UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Mobilní aplikace pro výživové poradenství

Jana Filipenská

Diplomová práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jana Filipenská**
Osobní číslo:	**I11373**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Mobilní aplikace pro výživové poradenství**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem práce je analýza, návrh a implementace mobilní aplikace pro výživové poradenství. Aplikace bude navržena tak, aby uživatel získal přehled o svém zdravotním stavu pravidelným zapisováním příjmu potravy do mobilní aplikace. Data budou zvoleným způsobem přenesena do báze dat webové aplikace, která bude zobrazovat statistiky a další doporučení. Mobilní plikace bude vyvinuta pro platformu Android.
V teoretické části budou popsány technologie pro vývoj mobilních a webových aplikací založených na jazyce Java. Budou porovnány používané operační systémy pro mobilní zařízení.
Implementační část se bude zabývat tvorbou samotné aplikace podle provedeného návrhu s využitím relační databáze a Java technologií. Analýza a návrh aplikace bude realizována s využitím jazyka UML.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

MURPHY L. M.: Android 2 - průvodce programováním mobilních aplikací. Comupeter Press, 2011.
MEIER R.: Professional Android 2 Application Development. Wrox, 2010.
UJBÁNYAI M.: Programujeme pro Android. Grada, 2012.

Vedoucí diplomové práce: **Ing. Zdeněk Šilar**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2012**
Termín odevzdání diplomové práce: **17. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2012

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 4. 5. 2013 Jana Filipenská

Poděkování

Ráda bych poděkovala svému vedoucímu diplomové práce Ing. Zdeňku Šilarovi za jeho čas, který věnoval konzultacím, jeho odborné rady a připomínky a možnost vytvářet tuto práci pod jeho vedením. Dále bych chtěla poděkovat všem, kteří mi poskytli rady důležité ohledně správné výživy a jak ovlivňuje nemocné lidi a za rady při programování.

Anotace

Tato diplomová práce se zabývá tvorbou aplikace pro výživové poradenství. Aplikace je dvojího druhu. Jedná se o webovou aplikaci a o aplikaci pro platformu Android. V mobilní aplikaci si uživatel zadává svůj denní příjem potravy. Tyto údaje poté pošle uživatel do webové aplikace, kde dojde k vyhodnocení a zobrazení statistik ohledně živin. Webová aplikace je tvořena pomocí technologie JSP a jako relační databázi využívá Oracle 10g Express Edition. Mobilní aplikace je napsána v Javě a využívá databázi SQLite.

Klíčová slova

Výživový poradce, Android, Java, JSP, SQLite, Oracle 10g Express Edition, živiny

Title

Mobile application for nutrition consulting.

Annotation

This thesis is focused on creating application for nutrition consulting. There are two types of applications. It is a web application and an application for the Android platform. In mobile application, users enter their daily food intake. These data are then sent by the user to the web application. Data are evaluated and the application displays statistics about nutrients. Web application is created using the JSP technology and as a relational database using Oracle 10g Express Edition. Mobile application is written in Java and uses the SQLite database.

Keywords

Nutrition consulting, Android, Java, JSP, SQLite, Oracle 10g Express Edition, nutrients

Obsah

Seznam zkratek

API	Application Programming Interface
BMR	Basal metabolic rate
CSS	Cascading Style Sheets
EE	Enterprise Edition
EJB	Enterprise JavaBeans
EL	Expression Language
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JDBC	Java Database Connectivity
JDK	Java Development Kit
JSF	JavaServer Faces
JSP	JavaServer Pages
JSTL	JavaServer Pages Standard Tag Library
ME	Micro Edition
MVC	Model–view–controller
OS	Operating Systém
SDK	Software Development Kit
SE	Standard Edition
SQL	Structured Query Language
UI	User Interface
XML	Extensible Markup Language

Seznam obrázků

Úvod

Tuto diplomovou práci jsem si vybrala na základě mnoha důvodů. Jedním z nich bylo, že z okolí znám pár lidí nebo kurzů, které se zabývají výživovým poradenstvím. Ať už z důvodu různých onemocnění, jako je např. cukrovka, kdy si ženy za účelem otěhotnění zaznamenávají svůj denní příjem potravy a sledují, jestli přijaly stejné množství energie kvůli dávkování inzulinu. Nebo z důvodu redukce váhy, kde se dá pomocí záznamů denního příjmu hlídat přijatou energii a rozložení živin a hlídat si tak úbytek váhy.

Avšak většina těchto způsobů zápisu denního příjmu potravy jsou vytvářeny pouze v papírové podobě. Kde si každý do tabulek přepisuje potravinu, kolik váží a na základě těchto údajů si sám vypočte počet živin a dodanou energii. A pár programů, které jsou na tuto problematiku zaměřeny, většinou pracují pouze v rámci jednoho dne a neumožňují uživateli zobrazit statistické výsledky, jak se mu daří plnit výživový plán nebo mají pouze malý výběr potravin a uživatel si nemůže přidat vlastní.

Dále je mi blízké programování v Javě a ráda sleduji vývoj mobilních technologií, kde je právě jeden z nejoblíbenějších operačních systémů Android od Googlu. A také bylo pro mě výzvou naučit se programovat pro v současnosti tolik oblíbený operační systém.

1 Vymezení cílů práce

Cílem práce je vytvořit mobilní a webovou aplikaci pro výživové poradenství. Díky nim uživatel získá přehled o svém zdravotním stavu pravidelným zapisováním příjmu potravy.

1.1 Cíle pro mobilní aplikaci

Mobilní aplikace bude určena pouze pro jednoho uživatele, vlastníka mobilního zařízení. Tuto aplikaci si stáhne z webových stránek. Práce s mobilní aplikací bude jednoduchá, přehledná, bude umožňovat snadnou manipulaci.

Jako první se uživateli zobrazí hlavní menu s tlačítky. Což bude představovat rozcestník mezi funkcemi a uživatel si tam bude moci zvolit, jestli si chce potraviny prohlížet, zadat svůj denní příjem potravin a jiné.

Nejdůležitější funkcí bude přidávání zkonzumovaných potravin. Takto přidané potraviny se uloží do seznamu dnů. A po zvolení konkrétního se zobrazí přehled denního příjmu potravin.

V menu prohlížení potravin budou potraviny řazeny podle kategorií, jako jsou sladkosti, pečivo, ovoce… Díky tomu bude pro uživatele vyhledání potraviny zcela intuitivní. Po vybrání konkrétní potraviny se mu zobrazí její konkrétní živiny a energetická hodnota s glykemickým indexem. Toto je užitečné pro to, aby uživatel získal přehled, zda je pro něj potravina, kterou chce zkonzumovat, vhodná.

Databáze potravin bude dostatečně obsáhlá. Avšak potravin je velké množství, proto aplikace bude umět přidávat novou potravinu, kterou uživatel v seznamu potravin nenašel. Také bude uživateli umožněno, aby své potraviny mohl v případě překlepu či chyby upravovat nebo také mazat. Nebude povoleno mazat systémové potraviny. Potraviny budou evidovány nejčastěji v gramech nebo mililitrech. Avšak při vytváření nové potraviny bude v nabídce i jiná jednotka, jako lžíce nebo lžička.

Další menu bude umožňovat poslat nově přidané a zkonzumované potraviny do webové aplikace. Ale pouze v případě, pokud uživatel bude mít správně vyplněné přihlašovací údaje. V opačném případě se mu zobrazí chybové hlášení s doporučením, aby si tyto přihlašovací údaje vyplnil.

1.2 Cíle pro webovou aplikaci

Aplikace bude určená pro více uživatelů. Uživatel se přihlásí pod svým přihlašovacím jménem a heslem. Také umožní registraci uživatele, který přihlašovací údaje ještě nemá. V registraci uživatel vyplní krom jména a hesla ještě další údaje, jako je váha, výška, věk, pohlaví a k jakému druhu diety bude aplikaci využívat. Na základě těchto údajů se budou počítat statistiky ohledně živin a energie. Po úspěšném přihlášení se jako první zobrazí úvodní stránka s možností stáhnutí mobilní aplikace.

Důležitou funkcí bude kalendář s odkazy na dny, kdy něco jedl. První vybraný je den, kdy uživatel naposledy jedl. Pod tímto kalendářem bude tabulka, kde se na každém řádku vypíše zkonzumovaná potravina v tomto dni, její množství, energetická hodnota a živiny. Na posledních řádcích se dané živiny sečtou a ukážou, jak si na tom uživatel stojí proti požadované optimální dietě. Tyto výsledky budou zvýrazněny pomocí sloupcového grafu, kde jeden sloupec bude představovat optimální množství a další sloupec skutečně zkonzumované množství.

Dále aplikace umožní zobrazovat statistiky ohledně poměru optimálního a skutečně přijatého množství živin a energetického příjmu. A to v rámci jednotlivých dnů, měsíců i roků.

1.3 Způsob dosažení cílů

Postup pro dosažení cílů práce bude následující. Nejprve si osvojím problematiku správné výživy pomocí knižních zdrojů a zjistím tak nezbytnou teorii o bílkovinách, tucích, sacharidech, glykemickém indexu a energii v potravinách. Dále vyhledám již existující řešení a porovnám je s plánovanou aplikací. Ještě před samotnou tvorbou aplikací provedu analýzu a návrh řešení problému, zvolím si prostředky pro vytváření aplikací a teprve poté začnu s implementací těchto aplikací.

2 Teoretický popis řešeného problému

Pro tvorbu programu zabývající se výživovým poradenstvím, je třeba nejdříve nastudovat teorii ohledně důležitosti živin, jako jsou bílkoviny, tuky, sacharidy, voda a energetické spotřeby člověka, která je u každého jedince rozdílná. Naučit se vypočítávat jejich optimální množství a uvědomit si, jak různá onemocnění toto optimální množství ovlivňují.

2.1 Bazální metabolizmus (BMR)

Jednou z důležitých sledovaných hodnot ve výživě je kolik osoba potřebuje přijmout energie za den. Ideálně by výdej energie měl být stejný jako příjem, jinak může dojít k úbytku na váze nebo naopak k jejímu zvyšování. Určení této potřebné denní energie je poměrně složité, každý je jedinečný a má jinou energetickou spotřebu. Například u dítěte a sportovce bude tato hodnota velmi rozdílná.

Článek (Kolik jídla bychom měli sníst, 2012) uvádí, že jednou z možností, jak zjistit tuto ideální energetickou hodnotu ve výživě je výpočet bazálního metabolizmu neboli BMR. Často se lze setkat právě s touto zkratkou a znamená Basal Metabolic Rate. BMR představuje optimální energetický příjem člověka. Nejpřesnější je tato hodnota pro člověka, který má průměrnou stavbu těla.

Na základě této hodnoty se tedy určí, kolik energie by měl uživatel v potravinách za den přijmout. Mělo by se však myslet na to, že hodnota BMR představuje základní množství energetického příjmu, které je třeba na pokrytí životních funkcí. Pokud uživatel například nepracuje v kanceláři, ale vykonává nějaké fyzicky náročné povolání, měl by si tuto hodnotu zvýšit. Nebo i při plánovaném sportu je třeba BMR zvýšit. Hodnota BMR je také nepřesná, pokud je uživatel velmi obézní nebo má velmi svalnaté tělo, tudíž nehodí se pro osoby s mimořádnou stavbou těla. V takovém případě je lepší konzultovat správnou výživu s diabetologem nebo obezitologem.

Hodnota bazálního metabolizmu se počítává podle věku, výšky, váhy a pohlaví. Rovnice pro výpočet BMR u muže a ženy jsou následující:

$$BMR1 = 66{,}473 + (13{,}7516 \times H) + (5{,}0033 \times V) - (6{,}755 \times R) \quad (1)$$

$$BMR2 = 655{,}0955 + (9{,}5634 \times H) + (1{,}8496 \times V) - (4{,}6756 \times R) \quad (2)$$

kde *BMR1* – bazální metabolizmus muže, kcal/den,
BMR2 – bazální metabolizmus ženy, kcal/den,
H – hmotnost, kg,
V – výška, cm,
R – věk, léta.

Příklad níže demonstruje výpočet BMR s konkrétními hodnotami. Vypočítává se BMR pro muže a ženu se při stejné výšce, váze a věku. Výsledek v kcal se pak převádí na kJ vynásobením o číslo 4,185.

$$BMR1 = 66{,}473 + (13{,}7516 \times 70) + (5{,}0033 \times 185) - \\ - (6{,}755 \times 29) = 1759\,\text{kcal}$$

$$BMR2 = 655{,}0955 + (9{,}5634 \times 70) + (1{,}8496 \times 185) - \\ - (4{,}6756 \times 29) = 1531\,\text{kcal}$$

Jak příklad ukázal, muž a žena vážící 70 kg, s výškou 185 cm a věkem 29 let mají velmi rozdílnou hodnotu BMR. Pro muže vychází na 1759 kcal (7361 kJ) a ženu 1531 kcal (6407 kJ). Počítá se totiž s tím, že muži mají všeobecně více svalové hmoty a tak při stejné práci spálí více kalorií než ženy.

Takto vypočtený energetický výdej je potřeba každý den přijmout. Pokud se plánuje redukce hmotnosti, doporučuje se denní příjem energie snížit o 2000 kJ. Nicméně u dospělého jedince by neměl energetický příjem klesnout pod 4186 kJ. Pro porovnání, průměrná hodnota BMR u žen je 6150 kJ, u mužů se sedavým zaměstnáním 9000 kJ a pro aktivní muže 10200 kJ.

2.2 Význam živin v potravě

Mezi nejdůležitější živiny patří bílkoviny, tuky, sacharidy. Tvoří 80 až 90 procent sušiny stravy, proto bývají označovaný jako hlavní. K bílkovinám patří také peptidy a aminokyseliny, ale mají v potravinách minimální výskyt. Organizmus se umí vyrovnat s krátkodobým nedostatkem živin tak, že si je sám vytváří. Ale pokud je nedostatek živin dlouhodobý, organizmus již není schopen si je déle nahrazovat a to vede k různým poruchám. Důležitostí a významem živin se zabývá (Pánek 2002) a z jeho publikace je zde čerpáno, pokud není uvedeno jinak.

2.2.1 Bílkoviny

Bílkoviny neboli proteiny patří mezi živiny, které není možné nahradit. Trávením se přeměňují na aminokyseliny a využívají se pro získávání energie, stavbu a obnovu tkání, svalů, buněk, kostí, chrupavek, vytváření enzymů, kreatinu, přepravují látky v těle. Představují tedy základní komponentu buněk organizmu důležitou pro imunitu. Využívají se jako zdroj energie, když má tělo nedostatek sacharidů a tuků. Tento stav není žádoucí, vzniká při drastických dietách, kdy tělo hladoví a dochází tak k úbytku svalové hmoty.

Nejlepším zdrojem bílkovin jsou živočišné bílkoviny, jako je maso, vejce, mléčné výrobky, protože obsahují největší podíl esenciálních aminokyselin. Bílkoviny mohou být také rostlinného původu. Jedná se o luštěniny, obiloviny, sóju. Mají však menší biologickou hodnotu než bílkoviny živočišného původu.

Pro určení ideálního příjmu bílkovin potřebných pro výživu, je třeba zvážit několik hledisek: celkovou potřebu bílkovin, obsah esenciálních aminokyselin, které si organizmus

nedokáže sám syntetizovat, poruchy metabolizmu, jako je celiakie, speciální nároky na příjem bílkovin u sportovců a u lidí s poruchou metabolizmu. Všeobecně však platí, že příjem proteinů by měl tvořit 10 až 15 procent energetického příjmu. Optimální je o něco vyšší množství než uvedené minimální, protože ne všechny aminokyseliny jsou vždy přítomny v ideálním množství.

2.2.2 Tuky

Tuky neboli lipidy mají ve výživě různou úlohu. Jsou největším zdroje energie ze všech uvedených živin. Jsou dvakrát vydatnější než sacharidy a bílkoviny. Představují zdroj lipofilních vitamínů, provitamínů a cholesterolu. Zlepšují jemnost chuti potravin, jejich konzistenci. Jejich kladem je, že vydolovávají pocit sytosti, ale teprve půl hodiny poté, co byla potravina zkonzumována a může tak dojít k situaci, že se tělu dodá více energie, než je třeba. Fungují tedy jako rezervní zdroj energie, chrání orgány, umožňují absorbovat vitamíny A, D, E, K.

Tuky se dělí na různé druhy a některé z nich jsou zdraví prospěšnější. Například nenasycené tuky patří mezi ty zdravější. Jedná se o rybí tuky, olivový olej, tuky v mořských plodech, sójový olej, oleje v ořechách. Jejich přínosem je, že snižují špatný cholesterol v těle a tak je menší riziko vzniku srdečních nebo cévních onemocnění. Tuky v másle, mase, palmovém oleji, sýrech patří mezi nasycené tuky a naopak zvyšují výskyt špatného cholesterolu. První, zdravější varianta tuků bývá při pokojové teplotě tekutá, nasycené tuky bývají spíše tuhé.

Aby skladba jídla byla vyvážená, tuky by měly tvořit 25 až 30 procent energetického příjmu. Podle článku (Tuky, 2012) je vhodné při stravování vynechávat máslo, slaninu, smetanu. Preferovat nízkotučné výrobky sýrů, zvýšit konzumaci ryb, hlavně mořských. Omezovat potraviny bohaté na cholesterol, jako jsou žloutky, vnitřnosti, tučné maso.

2.2.3 Sacharidy

Sacharidy neboli cukry představují nejdůležitější zdroj energie v těle. Jsou nepostradatelné pro některé buňky, jako jsou neurony nebo červené krvinky. Tělo je může při nedostatku vytvářet z aminokyselin a glycerolu. Jejich příjem je však nezbytný, aby se zabránilo snižování svalové hmoty při jejich nedostatku. Fungují tedy jako náhradní zdroj energie ve svalech a játrech.

Dělí se na monosacharidy, oligosacharidy a polysacharidy. Monosacharidy představuje hroznový cukr, ovocný cukr, med, tvoří je právě jedna cukerná jednotka. Díky tomu, že je není třeba dále štěpit, rychle se vstřebávají a dodávají tělu okamžitou energii. Je vhodné je konzumovat v rozumné míře, protože tělo po nich rychle vyhládne. Bývají označovány jako prázdné kalorie. A využívají je hlavně diabetici nebo sportovci pro okamžité dodání cukru.

Mezi oligosacharidy patří cukrová řepa, třtina, mléčný cukr, sladový cukr a jsou tvořeny dvěma až deseti cukernými jednotkami. Sacharidy tvořené více než deseti cukernými jednotkami se nazývají polysacharidy a patří mezi ně škrob, jejichž hlavním zdrojem jsou

obiloviny, brambory. Tyto sacharidy se vstřebávají pomalu a dodávají tak energii dlouhodobě. Všechny tyto sacharidy patří mezi dobře využitelné. Špatně využitelné sacharidy zastupují sladidla po diabetiky, luštěniny a další.

Vzhledem k tomu, že se jedná o nejdůležitější živinu v těle, měla byt tvořit 55 až 60 procent energetického příjmu.

2.3 Civilizační choroby a výživa

Další kapitola o výživě pojednává, jaké jsou specifické nároky na výživu u lidí, kteří trpí civilizačními chorobami, a popisuje jejich charakteristiky. Pojem civilizační choroby představuje podle článku (Civilizační choroba, 2013) nemoci, které se nejvíce objevují ve vyspělých zemích. Předpokládá se, že jsou důsledkem městského životního stylu. Příčinou jejich vzniku může být nadměrný stres, konzumace tučných a sladkých jídle, kouření, alkohol, nedostatek pohybu a další.

2.3.1 Obezita

Jedná se o nadměrný příjem energie. O obezitě se mluví, pokud podíl tukové tkáně na celkové hmotnosti přesáhne 20 procent u mužů, 25 procent u žen. Podle autorky (Müllerová, 2003) obezita zvyšuje riziko nádorových onemocnění, zkracuje délku života a zhoršuje jeho kvalitu. Doporučuje snížit váhu o 5 až 10 procent, snížit energetický příjem o 2000 kJ. Důležité je také dávat přednost potravinám rostlinného původu na úkor živočišných a zvýšit fyzickou aktivitu aerobního charakteru. Při snižování hmotnosti by měl být také snížen příjem jednoduchých sacharidů a naopak by se měl zvýšit příjem bílkovin. Nejlépe by měl být poměr živin tvořen z 30 procent tuků, 30 procent bílkovin a 40 procent sacharidů z celkového energetického přijmu.

2.3.2 Hypercholesterolémie

Podle článku (Vysoký cholesterol, 2012) hypercholesterolémie představuje zvýšené množství cholesterolu v těle. Cholesterol je důležitá látka pro tvorbu buněk, vytvoření vitamínu D, žlučových kyselin, produkci hormonů. Avšak pokud je ho nadbytek, ukládá se do stěny cév, díky tomu ztrácí pružnost a zmenšuje se její průměr. Snadněji tak může zachytit krevní sraženinu a může se stát, že některý orgán nebude mít dostatečný přívod krve a díky nedostatku kyslíku dojde k jeho odumření. Jedná se tedy o onemocnění, které se z počátku nijak neprojevuje, pouze zvýšenou hladinou cholesterolu. Ale jeho nebezpečným následkem je pak jeho ukládání do stěn cév a zvýšené riziko srdečního infarktu.

Prevencí proti vysokému cholesterolu bývá správná životospráva. Příjem tuků by rozhodně neměl přesáhnout 30 % celkové energie. Také by se mělo dávat přednost nenasyceným kyselinám před nasyceným. Dále je třeba si hlídat váhu, dodávat tělu více vlákniny, zeleniny, omezovat pečení a smažení, zvýšit pohybovou aktivitu.

2.3.3 Hyperlipidémie

Podle autora (Štulc, 2001) se jedná o onemocnění cévní stěny, které může vést k cévní mozkové příhodě, ischemické chorobě srdeční a dalších. Znakem tohoto onemocnění je zvýšené množství lipoproteinů v krvi. Mezi viditelné příznaky patří hromadění tuků na kůži víček, nad šlachami končetin, v očích a bolestmi břicha. Většinou je to způsobeno dědičnou poruchou. Postihuje až 20 % populace, jen nebývá dlouho viditelná. Léčba hyperlipidémie je zaměřená na dietní opatření. Zásadou je snížit cholesterol. Příjem sacharidů by měl tvořit 55 % energie, bílkovin 30 % energie a tuků maximálně 30% energie. Mastné kyseliny by měly být tvořeny z 1/3 nasycených kyselin, z 1/3 mononenasycených kyselin a z 1/3 polynasycených.

2.3.4 Diabetes mellitus

Diabetes mellitus neboli cukrovka má dva typy. První typ je závislý na inzulinu. V těle nedochází k jeho dostatečné tvorbě a tak musí být dodáván podkožně. Z celkového počtu diabetiků touto chorobou trpí pět až deset procent. Nejčastěji se začíná objevovat mezi desátým až patnáctým rokem. Při léčbě této nemoci by se mělo pamatovat na správnou životosprávu. Měl by se zvýšit podíl vlákniny ve stravě. Tělu by se mělo dodat takové množství sacharidů, které odpovídá energetické potřebě, tělesné hmotnosti a množství aplikovaného inzulinu. Strava by měla být podávána pravidelně s konstantním poměrem živin.

Diabetes mellitus druhého typu je nejčastější. Tento typ se vyskytuje po čtyřicátém roce. Nejvíce však u lidí, kteří trpí obezitou. Kvůli nedostatku inzulinu se málo využívá krevního cukru a tak je hladina tohoto krevního cukru zvýšená. Díky špatnému uvolňování inzulínu dochází k poruše jeho působení v cílových tkáních.

U tohoto typu cukrovky se doporučuje snížit tělesnou hmotnost a množství cukrů v potravě a zmenšit tak celkový příjem energie. Zásadou však u obou typů je rozdělit stravu na více malých porcí. Konzumovat potraviny s nižším glykemickým indexem. Sacharidy by měly tvořit 50 až 55 procent energetického příjmu, tuky 30 procent a bílkoviny 15 až 20 procent. Alkohol by se měl pít pouze ve výjimečných případech, spíše vůbec a to z důvodu vzniku hypoglykemie.

2.3.5 Dna

Dna představuje kloubní onemocnění. O této problematice pojednává článek (Dieta při kloubním onemocnění, 2012). Dna se vyznačuje zvýšenou hladinou kyseliny močové a lze ji dietou velmi dobře ovlivnit. Projevuje se otoky, bolestmi, nejčastěji u kloubů palců, záněty žil, bolestmi břicha. Jedná se o poruchu přeměny bílkovin, kdy se zvyšuje počet močanů a ty se ukládají do chrupavek u kloubů nebo jako močové kameny v ledvinách.

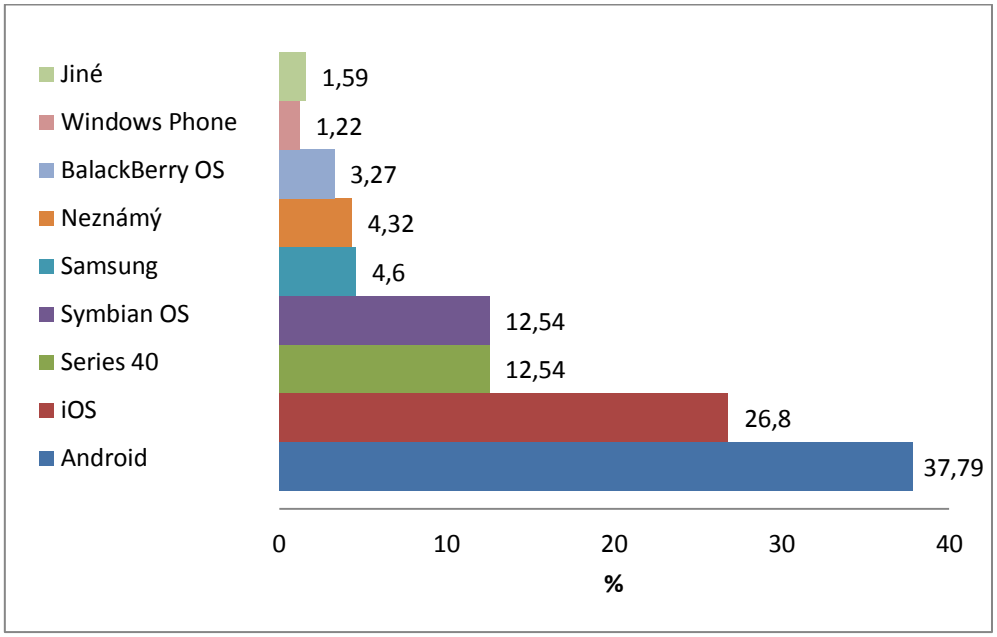
Při dietě se doporučuje snížit příjem bílkovin na 50-60 g/den, což představuje 0,6-0,8 g/kg váhy. Maso by se mělo konzumovat jednou denně a to dušené ve vodě, aby došlo k odplavení purinů. Uzená masa, zvěřina, vnitřnosti, černé čaje, čokoláda by se měly ze stravy úplně vyloučit. Šetřit by se také mělo i s dráždivými druhy koření. Dále je vhodné zvýšit množství tekutin, pomáhají odplavovat z těla přebytečnou kyselinu močovou.

3 Operační systémy pro mobilní zařízení

Problematiku operačních systémů pro mobilní zařízení dobře popisuje článek z Wikipedie (Mobile operating systém, 2013). Pokud není uveden jiný zdroj, následující oddíly vychází z tohoto článku a popisují hlavní vlastnosti nejpoužívanějších operačních systémů a na závěr se zhodnotí vhodnost systémů pro vývoj aplikací.

3.1 Využití jednotlivých operačních systémů

Následující obr. 1 je převzatý z článku webové stránky StatCounter (Top 8 Mobile Operating Systems form Mar to Arp 2013, 2013). Je na něm znázorněno procentuální využití mobilních operačních systémů. Z obrázku vyplývá, že mezi nejpoužívanější patří Andoroid, iOS, Series 40 a SymbianOS.



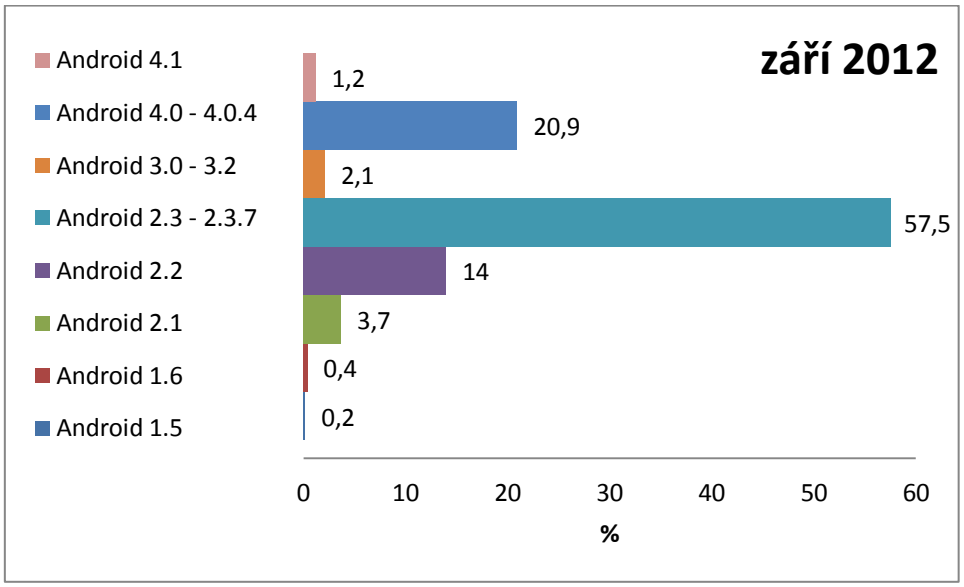
Obrázek 1 – Rozšíření mobilních operačních systémů

3.2 Android

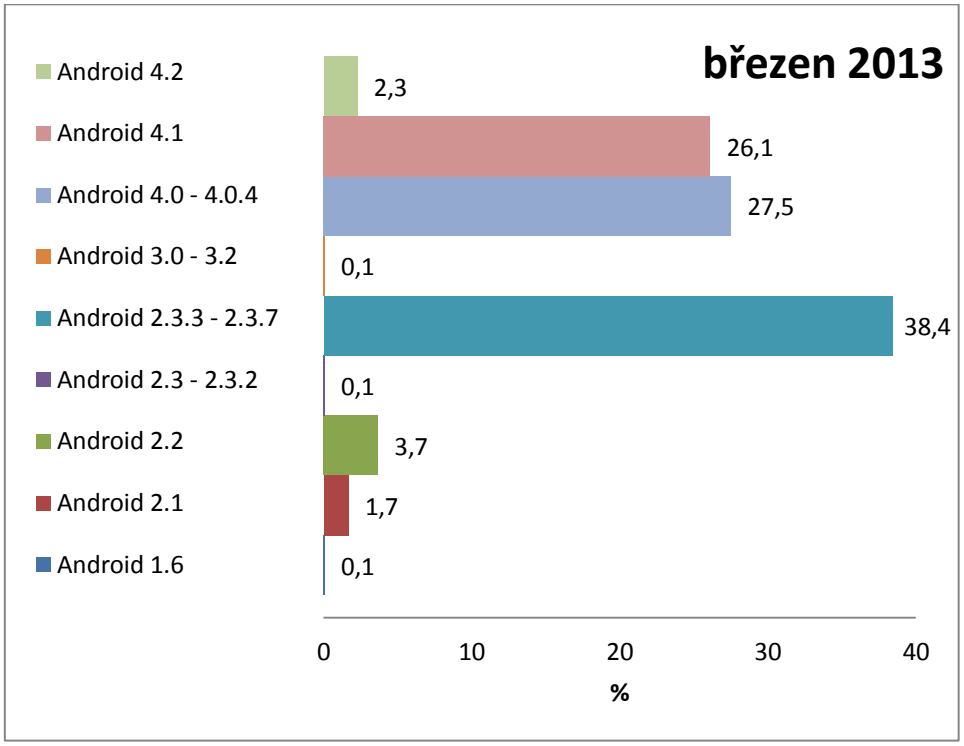
Jedná se o operační systém od Google Inc., kterou převzal od společnosti Android Inc. Podle autora (Ujbányai, 2012, s. 13) je systém založený na open source platformě. To znamená, že jde užívat bezplatně a přistupovat ke zdrojovým kódům. Aplikace nepřistupují k funkcím jádra přímo, ale přes Android API. Jádro Androidu může běžet na rozdílném hardwaru. Představen byl koncem roku 2007 a ihned byl mezi vývojáři oblíben. Do verze 2.0 byl určen spíše pro mobilní zařízení. Další verze 2.x jsou určeny již jak pro mobilní zařízení, tak i tablety. Android 3.0 je určen pouze pro tablety. Zvláštností těchto systémů jsou jejich názvy podle cukrovinek a dezertů: Cupcake (1.5), Honeycomb (3.0), Jelly Bean (4.1)

Následující obrázky zobrazují procentuální zastoupení jednotlivých verzí pro platformu Android. První statistika je ze září 2012 od autora (Wells, 2012), kdy začala být mobilní

aplikace vytvářena a byla tedy rozhodující pro výběr minimální a cílové SDK verze. Pro porovnání je uvedena další statistika z března 2013 z webové stránky Android Developers (Dashboards, 2013).



Obrázek 2 – Procentuální zastoupení verzí platformy Android (září 2012)



Obrázek 3 – Procentuální zastoupení verzí platformy Android (březen 2013)

3.3 iOS

Systém iOS je od společnosti Apple Inc. Jedná se o uzavřený systém, který umožňuje instalovat aplikace pouze přes App Store. Je určen pro mobilní telefony iPhone a jiná

mobilní zařízení: iPod Touch, iPad. Podle článku (IOS (Apple), 2013) jdou pouze instalovat aplikace napsané v jazyce C nebo Objecite-C schválené společností Apple.

3.4 Series40 (S40)

Stále ještě velmi oblíbený je systém Series40 od společnosti Nokia. Je určený spíše pro starší jednodušší mobilní zařízení. V průběhu let běžel na více jak 150 telefonních modelech.

3.5 Symbian OS

Jedná se o systém od společnosti Nokia a Accenture s otevřenou licencí. Kolem roku 2009 byl velmi populární. Jeho podíl na trhu byl přes 50 %. Je používán v řadě mobilních zařízení, jako jsou Nokia, Samsung, Sony Ericsson a v dalších. V současnosti ale podle předchozího obr. 1 jeho podíl na trhu výrazně klesá.

3.6 BlackBerry 10

BlackBerry 10, dříve BlackBerry BBX, je systém vyvíjený společností BlackBerry a představuje další generaci platformy pro BlackBerry chytré telefony a tablety. Jak je na obr. 1 ukázáno, systém nepatří mezi nejvíce oblíbené, ale i přesto si vede lépe než systém Windows Phone.

3.7 Windows Phone

Windows Phone vyvíjí společnost Microsoft. Představuje nástupce systému Windows Mobile. Systém obsahuje nové uživatelské rozhraní zvané Metro skládající se z dlaždic. Dlaždice představují aplikace, mediální položky a další. Podporuje plnou integraci služeb od Microsoftu, jako je Microsoft SkyDrive a Office Mobile, Xbox Music, Xbox video, Xbox Live hry a samozřejmě vyhledávač Bing. Dalšími integrovanými službami třetích stran jsou Facebook a Google účty. Aplikace TellMe dovoluje ovládat systém hlasem. Jádro Windows Phone 8 vychází z řady Windows NT místo architektury Windows CE.

3.8 Hodnocení

Z předchozích oddílů vyplývá, že operační systémy Symbian OS a Series40 patří stále mezi nejpoužívanější čtyři systémy, ale dá se předpokládat, že jejich podíl na trhu bude dále klesat a tak vývoj aplikací pro tyto systémy nemusí být vhodný.

Mezi zbylé nejoblíbenější systémy patří Android a iOS. Podle Wikipedie (Comparison of mobile operating systems, 2013) na iOS je nepříjemné to, že je třeba pro vývoj Mac OS a stojí $99 za rok, ale naproti tomu se neplatí poplatek za zveřejnění aplikace na App Store. U aplikace pro Android se sice za vývoj neplatí nic, ale zveřejnění aplikace na Google Play stojí $25. Vzhledem k tomu, že aplikace pro Android se ale nemusí stahovat z oficiálního obchodu, ale z kterékoliv stránky na internetu, nejvýhodněji vypadá vývoj aplikací pro Android.

4 Technologie pro vývoj aplikací založených na jazyce Java

Tato kapitola popisuje nejznámější technologie pro tvorbu webových a mobilních aplikací v jazyce Java. Vysvětluje rozdíly mezi těmito technologiemi. A navrhuje, pro jaké aplikace je vhodné je použít.

4.1 Technologie pro vývoj webových aplikací

Problematiku technologií určených pro vývoj webových aplikací v jazyce Java nejlépe vystihuje autorka Dana Nourie (Nourie, 2006). Nejprve je třeba definovat, co to je webová aplikace. Jedná se o aplikaci dostupnou z webového serveru pomocí internetu. Webový prohlížeč zde představuje klienta. Důvodem její velké oblíbenosti je, že uživatelé nemusí instalovat žádný software.

Java technologie, které jsou používané k vytváření webových aplikací, jsou součástí Java EE platformy. Jejím základem je platforma Java SE. Aby tyto aplikace mohly být na serveru spuštěny, musí mít server nainstalován Java server software.

V následujících oddílech jsou popsány nejčastěji používané technologie.

4.1.1 Java Servlet API

Servlet API je třída definující standardní rozhraní pro obsluhu požadavků a odpovědí mezi klientem a webovým serverem. Požadavky jsou obsluhovány přes toto rozhraní. Obsahuje balíčky. Jeden balíček `javax.servlet` obsahuje rozhraní pro implementaci různých servletů pro různé protokoly jako je HTTP, HTTPS, FTP. Pro využívání HTTP protokolu je třeba vložit balíček `javax.servlet.http`.

Z popisu vyplývá, že tato technologie je využívána všude tam, kde je třeba obsluhovat požadavky a odpovědi.

4.1.2 JavaServer Pages (JSP)

Technologie JSP umožňuje vytvářet dynamický obsah webových stránek. Umožňují vytvářet aplikace, které jsou nezávislé na serveru a platformě. Obdobně jako u PHP se mezi speciální znaky <% %> píše Java kód. Tyto stránky mají příponu .jsp. Servletový kontejner je převádí na servlety, pokud se uživatel pokusí poprvé načíst stránku a dále je překládá do Java třídy. Pokud se v .jsp souboru upraví kód, je stránka opět převedena na servlet a znovu zkompilována. Další zobrazování JSP stránky jsou velmi rychlá, protože se nemusí znovu analyzovat. Následující zdrojový kód je ukázka JSP stránky z webové aplikace.

```
<h2>Přihlášení</h2>
<% if(request.getAttribute("spravneZadaneHeslo")== "false"){%>
<p>Zadali jste špatně e-mail nebo heslo.</p>
<% } %>
```

Z přechozího odstavce vyplývá, že psaní Java kódu mezi znaky <% %> se projeví rychlejším vývojem stránek. Má však také jednu nevýhodu: nepřehlednost.

4.1.3 JavaServer Pages Standard Tag Library (JSTL)

Zapouzdřuje základní funkce společné mnoha JSP aplikací. Místo toho, aby se mixovaly různé značky, díky JSTL se dá používat sada standardních značek. JSTL má značky pro práci s proměnnými, cykly, větvením, značky pro přístup k databázi použitím SQL, pro práci s XML. Formátovací značky umožňují zobrazovat časové údaje, nastavit formát číslům.

JSTL se využívá společně s jazykem EL. Jak uvádí článek (JavaServer Pages, 2013), prostřednictvím tohoto jazyka se přistupuje k datům z aplikace, requestu nebo session. Zapisuje se do speciálních znaků ${} a umisťuje se do JSP stránek.

4.1.4 JavaServer Faces (JSF)

Jedná se UI framework pro vytváření webových aplikací. Hlavní komponenty JSF technologie obsahuje framework pro GUI komponenty, flexibilní model pro vykreslování komponent v různých značkovacích jazycích a standardní `RenderKit` pro generování kódu HTML.

Klady a zápory technologie dobře popisuje článek (JavaServer Faces, 2013), kde mezi jejich hlavní výhody patří, že jsou součástí Javy EE, má dostatek dokumentace. A mezi nevýhody patří nutnost přidat Seam a Facelets frameworky. Využívání speciálních funkcí této technologie je neefektivní a složité.

4.1.5 JDBC API

JDBC API umožňuje vytvářet databázové SQL příkazy pomocí programovacího jazyka Java. Pokud je třeba přístup k relační databázi, dá se použít v servltech, JSP stránkách i v EJB. Důležitou součástí této technologie je JDBC ovladač.

Výhodou této technologie je, že může být použita v řadě jiných Java technologií a že aplikace může být vytvořená pro jakékoliv databáze, které využívají stejný SQL jazyk.

4.1.6 Java Persistence API

Jedná se o standard Javy EE. Pomocí objektově relačního mapování zajišťuje perzistenci dat. Java technologie pro zajištění perzistence se skládá z Java Persistence API, dotazovacího jazyka a objektově relačních mapovacích metadat.

4.1.7 Enterprise JavaBeans (EJB)

Poslední technologii dobře popisuje článek (Enterprise JavaBeans, 2013). EJB představuje základní komponentní architekturu. Jedná se o objekty představující aplikační logiku systému a vytváří je vývojář. Používá se u tří a více vrstvých aplikací.

Tato technologie se využije hlavně, pokud je v aplikaci potřeba současný přístup k datům a transakce, pokud existuje více různých typů klientů. Avšak je těžší na naučení, má jistou režii. Nedoporučuje se používat, pokud nejsou potřeba vlastnosti EJB.

4.1.8 Hodnocení

Z vyjmenovaných technologií nejlépe vypadá použití kombinace Servletů, JSP stránek nebo JSF, pokud nebude třeba využívat speciální funkce. JSTL společně s jazykem EL může kód ve stránce JSP zpřehlednit.

Pokud bude aplikace přistupovat k databázi, je nutností využívat JDBC API společně s ovladačem. Také by se mělo zvážit použití Java Persistence API společně s EJB a např. s frameworkem JSF, pokud bude aplikace přistupovat do relační databáze.

Technologie EJB je vhodná, pokud se jedná o tvorbu podnikové aplikace. V jednoduché uživatelské aplikaci její využití ztrácí smysl a je možné použít technologií JSP, Servletů a JavaBean.

4.2 Technologie pro vývoj mobilních aplikací

Při programování aplikací pro mobilní aplikace se využívají např. tyto technologie: Java Development Kit, Android Software Development Kit, Java ME. V práci (Ujbáyai, 2012, s. 23-25) jsou oba vývojové nástroje velmi dobře popsány.

4.2.1 Java ME

Jak uvádí článek (Java Platform, Micro Edition, 2013), jedná se o platformu nabízející API pro vývoj software pro malá zařízení nebo pro zařízení s omezenými prostředky, jako je paměť, displej, kapacita baterie.

Java ME platforma je open source řešení pro vytváření mobilních aplikací. Umožňuje nezávislost aplikace na platformě.

Je velmi oblíben u operačního systému Series40 od společnosti Nokia. Využíváno v systému Bada i na Symbian OS. Také jsou tu implementace pro Widows CE, Windows Mobile, Maemo a MeeGo. Avšak některé nejnovější platformy tuto technologii nepovolují. Jedná se o iPhone, Windows Phone, BlackBerry 10, Android.

4.2.2 Java Development Kit (JDK)

Java Development Kit obsahuje důležité knihovny a nástroje pro tvorbu aplikací a apletů pro platformu Java. Je nezbytnou součástí pro programování všech aplikací napsaných v jazyce Java. Obsahuje Java Runtime Enviroment, bez něj by nešlo aplikaci spustit, dále debugger, ovladač a další.

4.2.3 Android Software Development Kit (SDK)

Software Development Kit slouží jako sada vývojových prostředků umožňující tvorbu aplikací pro herní konzole, hardwarové platformy a operační systémy. Obsahuje knihovny Javy pro vytváření rychlých mobilních aplikací pro Android, dokumentaci, ukázky zdrojových kódů a emulátor neboli virtuální mobilní zařízení fungující na běžném počítači.

Podle článku (Andorid software development, 2013) Android nepoužívá zavedené Java standardy, jako je Java SE a Java ME. Díky tomu nejsou kompatibilní Java aplikace

napsané pro tyto platformy s platformou Android. Android pouze využívá z Javy syntaxi a sémantiku, ale neobsahuje všechny knihovny a API používané v Java SE nebo ME. Nicméně na trhu existuje několik nástrojů, které umožňují převést Java ME na Android.

4.2.4 Hodnocení

Java ME je tedy nejvhodnější pro tvorbu jednoduchých aplikací využívaných u starších mobilních zařízení, která nemusí mít operační systém a je možné ji převést na jinou platformu.

Pokud je třeba vytvořit složitější aplikaci, která poběží na platformě Android, musí se použít SDK. Nevýhodou těchto aplikací je nepřevoditelnost na ostatní platformy.

4.3 Použité technologie

Tento oddíl se zabývá vybráním nejvhodnější technologie pro tvorbu aplikací na základě jejich vlastností popsaných v předchozích oddílech.

4.3.1 Technologie použité ve webové aplikaci

Pro tvorbu webové aplikace bylo z výše uvedených technologií použito Java Servlet API společně s JSP. Pro lepší přehlednost kódu v JSP stránce byla dodržována architektura MVC. Kde JSP stránky představují view, Servlet představuje controller a ten volá pomocné třídy představující model.

Jelikož webová aplikace přistupuje k databázi, bylo třeba použít technologii JDBC API s ovladačem. Jako databáze byla vybrána Oracle Database 10g Express Edition. Jak uvádí článek (Oracle Documentation 10g Release 2, 2013), jedná se o bezplatnou verzi, je snadná na instalaci, ovládání a vývoj. Pracuje se s ní prostřednictvím prohlížeče a umožňuje spravovat databázi, vytvářet databázové objekty, importovat a exportovat data, provádět dotazy a SQL skripty aj.

Pro analýzu a zobrazení statistik spotřebovaného jídla byla vybrána technologie Google Chart Tools. Poskytuje mnoho druhů přehledných grafů a tabulek, které jsou snadno upravitelné. Tabulky i grafy jsou tvořeny skriptovacím jazykem JavaScript, tudíž není třeba žádný zásuvný modul nebo knihovna.

4.3.2 Technologie použité v mobilní aplikaci

Aplikace pro výživové poradenství je složitějšího charakteru a vzhledem k tomu, že mobilní zařízení se systémem Android jsou v současné době nejrozšířenější, aplikace byla vytvořena právě pro tuto platformu.

V mobilní aplikaci je třeba používat databázi, ale pouze takovou, která je vhodná pro zařízení s malou kapacitou paměti. V práci (Meier, 2010, s. 38-41) jsou popsány prostředky, které Android poskytuje. Android nabízí odlehčenou relační databázi pro každou aplikaci. Zaručuje i bezpečnost dat, v základním nastavení její obsah může vidět pouze aplikace, která ji vytvořila. Z těchto důvodů byla vybrána právě databáze SQLite.

5 Existující řešení

Při tvorbě aplikace je dobré znát konkurenční řešení. Vyhledat, v čem podobné aplikace vynikají nebo naopak, kde mají slabiny nebo jestli vůbec existují. Při vývoji tak lze předejít chybám, které udělali autoři konkurenčních aplikací, případně se z nich inspirovat a dodat tak další vhodné funkce. Průzkum stávajících aplikací je popsán v této kapitole.

5.1 Webové aplikace

Na internetu není mnoho webových aplikací, které by se zabývaly výživovým poradenstvím. Všeobecně se jedná o aplikace nabízené zdarma. Hlavní zástupci těchto aplikací jsou spolu s jejich vlastnostmi uvedeny níže.

5.1.1 E-deník

Tato aplikace se nachází na adrese http://www.priteltvehosrdce.cz. Před začátkem používání se uživatel musí zaregistrovat. Při registraci vyplní údaje o své váze, výšce a životním stylu.

Aplikace slouží k sledování příjmu a výdeje energie. Uživatel zadává nejen potraviny, které snědl, ale i fyzickou činnost, kterou v dané dny provozoval. Aplikace poté sleduje poměr příjmu a výdeje energie a vše zobrazuje v přehledných grafech a tabulkách.

Aplikace obsahuje asistenta pro redukci váhy, který po zadání cílové váhy, rychlosti hubnutí vytvoří přehledný plán.

5.1.2 Kalorie MTE

Tato aplikace se nachází na adrese http://kalorie.mte.cz. Uživatel si před použitím musí vytvořit účet a zadat o sobě všechny potřebné údaje. Uživatelské rozhraní je rozděleno do třech hlavních částí (jídlo, aktivity, můj). Každá část je pro přehlednost laděna do jiné barvy.

V části s názvem jídlo, která je laděná do zelené barvy, uživatel zadává informace o potravinách, které snědl. Je zde dostupná rozsáhlá databáze jídel. Pokud se v ní jídlo nenachází, je možno přidat vlastní. Snědená jídla jsou zobrazována v přehledné tabulce. V části s názvem aktivity, která je laděná do oranžové barvy, se zadává fyzická aktivita, kterou uživatel v jednotlivých dnech vykonával. Je zde velké množství přednastavených aktivit, které může uživatel použít. Nechybí zde možnost přidat svou vlastní aktivitu.

V části s názvem můj účet, která je laděná do modré barvy, může uživatel nastavovat informace o své osobě. Jsou zde tabulkové přehledy energetických výdejů a příjmů. Lze zde také vygenerovat grafy jednotlivých veličin, jako například poměr výdeje a příjmu energie, nebo vývoje tělesné hmotnosti.

5.1.3 Nutriční kalkulačka

Tato aplikace je dostupná na adrese http://lecbacukrovky.cz/nutricni-kalkulacka. Aplikace je velmi jednoduchá a neobsahuje žádnou možnost registrace. Jedná se o nutriční

kalkulačku na jeden den. Středem aplikace jsou záložky, kde každá ze záložek tvoří jeden denní chod. Na záložce uživatele vybere potravinu z databáze dostupných potravin, nebo zadá svou vlastní.

Ve spodní části je přehledová tabulka, která obsahuje souhrnný přehled všech potravin, zadaných přes jednotlivé záložky. Jsou zde zobrazeny všechny důležité parametry, týkající se výživy. Podle typu diety, kterou uživatel zvolil, se zde zvýrazní, zda snědené potraviny této dietě vyhovují.

5.1.4 Hodnocení

První dvě zmiňované aplikace nabízejí velice podobnou funkcionalitou. Obě nabízejí možnost zadávat po dnech příjem potravin a fyzickou aktivitu. Nevýhodou první je nepřehledné a neintuitivní uživatelské rozhraní. Uživatelské rozhraní je naopak velikou výhodou druhé aplikace, u které je velice povedené a přehledné.

Třetí aplikace nabízí ve srovnání s prvními dvěma jen velice omezené možnosti. Má poměrně povedené a přehledné uživatelské rozhraní, ale jelikož se zde uživatel nemůže přihlásit a ukládat zkonzumované potraviny, hodí se tato webová aplikace spíše pro plánování budoucích jídelníčků.

5.2 Mobilní aplikace

Android Market nabízí velké množství mobilních aplikací pro výživové poradenství. Většinou ale neobsahují češtinu a bývají různým způsobem zpoplatněny. Nejzajímavější mobilní aplikace, zabývající se touto problematikou, jsou popsány v následujících pododdílech.

5.2.1 Diet Assistant

Aplikace je nabízena v placené a neplacené verzi. Hlavní přidaná hodnota placené verze je, že si v ní lze vytvářet vlastní dietní plány. Součástí neplacené verze je šestnáct dietních plánů, které vyhoví většině uživatelů. Jsou zde plány založené na konzumaci ovoce, zeleniny, mořských plodů, ale i plány s pestrou stravou složené z širokého spektra potravin. Každý dietní plán trvá sedm dnů a jeho cílem je směřovat ke stanovené váze, kterou si uživatel před započetím diety v aplikaci nastaví.

Dietní plán je rozdělen na sedm obrazovek, kde každá obrazovka představuje jeden den diety. Každý den je rozdělen do pěti chodů (snídaně, dopolední svačina, oběd, odpolední svačina, večeře). Chod je tvořen jedním, nebo několika jídly. Pokud některé jídlo uživateli nevyhovuje, může si za toto jídlo vybrat náhradu z několika alternativ, které aplikace pro každou položku v dietním plánu nabízí.

V průběhu diety může uživatel postupně zadávat vývoj své hmotnosti a aplikace dokáže z těchto dat kreslit graf vývoje hmotnosti v čase. Aby uživatel nezapomínal svou váhu aktualizovat, je zde možnost nastavit si připomínání ve stanovených časech.

Užitečnou funkcí je, že aplikace dokáže pro dietní plán vytvořit přehledný nákupní list. V nákupním listu lze jednotlivé položky odškrtávat a tím je označovat jako již zakoupené.

V aplikaci je také integrované diskuzní fórum, s jehož pomocí si uživatelé mohou vyměňovat zkušenosti a navzájem si radit nebo mohou zveřejnit své úspěchy na sociálních sítích.

5.2.2 Diet Point - Weight Loss

Pro přístup do aplikace je nejdříve nutné se registrovat. Registraci lze pohodlně provést přímo v aplikaci, nebo na webových stránkách aplikace.

V základu je obsažené velké množství dietních plánů. Jen část z nich je však dostupná v bezplatné verzi aplikace. Dietní plány jsou uzamčeny a přístup k nim uživatel dostane až po zakoupení plné verze aplikace. V bezplatné verzi také nelze vytvořit vlastní plán. Dietní plány navíc lze procházet, pouze pokud je k dispozici připojení k internetu.

Dietní plán je pro každý den rozdělen do pěti chodů. Není však příliš pružný a nenabízí pro jednotlivé chody různé alternativy, ale vždy pouze jednu možnost.

V aplikaci lze zadávat průběžně vývoj své váhy. Tato data jsou přenášena do webové části aplikace, kde se zobrazují v přehledném grafu. Součástí aplikace je také nástroj pro sestavování nákupního seznamu potravin, tento je však dostupný až v placené verzi.

5.2.3 Calorie Counter

Prvním krokem při používání aplikace je registrace. Během ní uživatel zadá základní údaje, jako například váhu, výšku a stanový cíl pro hubnutí (datum a váhu). Aplikace pro jednotlivé dny ukazuje přehlednou tabulku živin, ve které je uvedeno, kolik které složky má uživatel za daný den sníst a kolik již snědl.

Snědené potraviny se přidávají do aplikace pomocí přehledného rozhraní. Uživatel vybere, o které denní jídlo se jedná a poté v databázi vyhledá patřičnou potravinu. Pro snadnější hledání je možné vyhledat potravinu pomocí čárového kódu s jejího obalu, který lze načíst pomocí vestavěného fotoaparátu. Pokud některá potravina v databázi potravin není, lze jí do této databáze přidat.

Na webových stránkách aplikace je po přihlášení možné prohlížet veškerá data, týkající se uživatelova profilu, snědených potravin a zadávat snědené potraviny.

5.2.4 Hodnocení

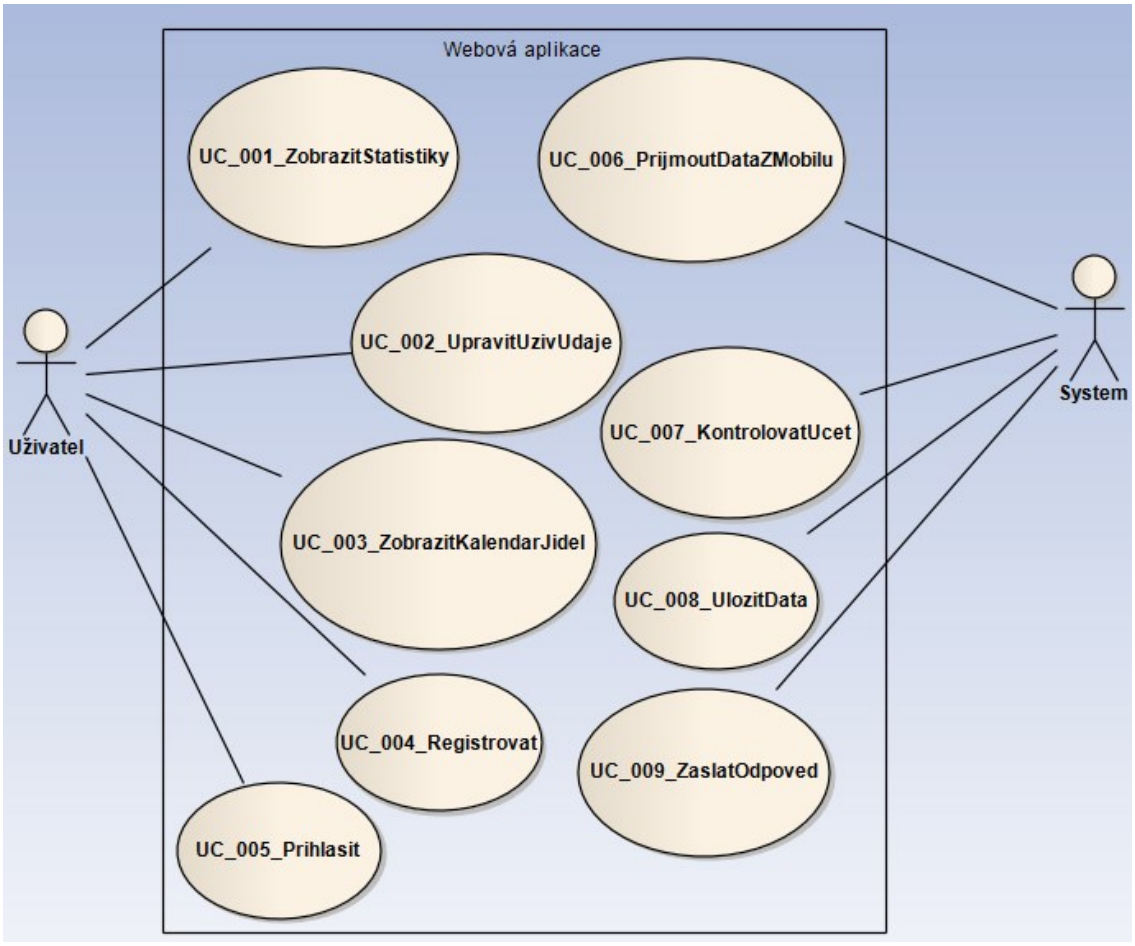
Všechny aplikace nabízejí velice podobnou základní funkcionalitu. První dvě zmiňované aplikace jdou cestou dietního plánu, který stanovuje, jaké potraviny se budou v který moment konzumovat. Poslední zmiňovaná aplikace žádný plán neobsahuje, ale pouze zaznamenává snědené potraviny. Všechny aplikace dokáží vytvářet různé statistiky, které se týkají nutričních hodnot snědených potravin v časovém období.

Nejméně použitelná se zdá druhá aplikace. Její uživatelské rozhraní není příliš intuitivní. Navíc v dietním plánu nelze pro daný chod vybírat mezi alternativou z více jídel.

6 Definice požadavků na aplikaci

V této kapitole jsou představeny diagramy případů užití, které specifikují, co mají aplikace dělat, jaké jsou jejich funkce. Tyto diagramy vycházejí z cílů práce. Nejprve bylo třeba nalézt hranice systému, aktéry a určit případy užití.

6.1 Use Case diagram webové aplikace



Obrázek 4 – Diagram případů užití webové aplikace

V tomto diagramu se nachází dva aktéři. Jelikož se jedná o uživatelskou aplikaci, jedním z aktérů je tedy uživatel. Dalšího aktéra představuje systém této webové aplikace, který má za úkol přijímat data z mobilní aplikace. Dále jsou rozebrány jednotlivé případy užití a seřazeny podle pořadí přístupu k aplikaci.

UC_005_Prihlasit: Uživatel zadá přihlašovací údaje. Systém ověří údaje uživatele. V případě, že jsou správně, přesměruje uživatele na uvítací stránku. Jinak systém zobrazí chybovou zprávu.

UC_004_Registrovat: Systém vytvoří nový účet uživatele. Případ je spuštěn tlačítkem „Registrovat“. Dokud jsou přihlašovací údaje neplatné, systém žádá o jejich správné vyplnění. Nakonec systém vytvoří účet.

UC_003_ZobrazitKalendarJidel: Případ je spuštěn uživatelem, vybráním příslušné položky v menu. Systém zobrazí kalendář se zvýrazněnými políčky, kdy uživatel jedl. Pod kalendářem systém zobrazí tabulku jídel a graf s optimálním a skutečným množstvím živin.

UC_002_UpravitUzivUdaje: Případ užití začíná, až uživatel vybere ve formuláři „Upravit“. Systém zkontroluje vyplněné údaje. Pokud údaje nejsou správné, informuje jej o tom chybovým hlášením tak dlouho, dokud údaje nejsou správné. Systém uloží nová data.

UC_001_ZobrazitStatistiky: Uživateli se zobrazí statistiky ohledně stravování. Případ užití začíná po výběru položky v menu. Když zákazník vybere „Statistické údaje živin“, systém zobrazí grafy s optimálním a skutečně dodaným množstvím živin v rámci dnů, měsíců i roků. Pokud zákazník vybere „Statistické údaje BMR“, systém zobrazí stejný graf jako u živin, ten se však týká dodané energie.

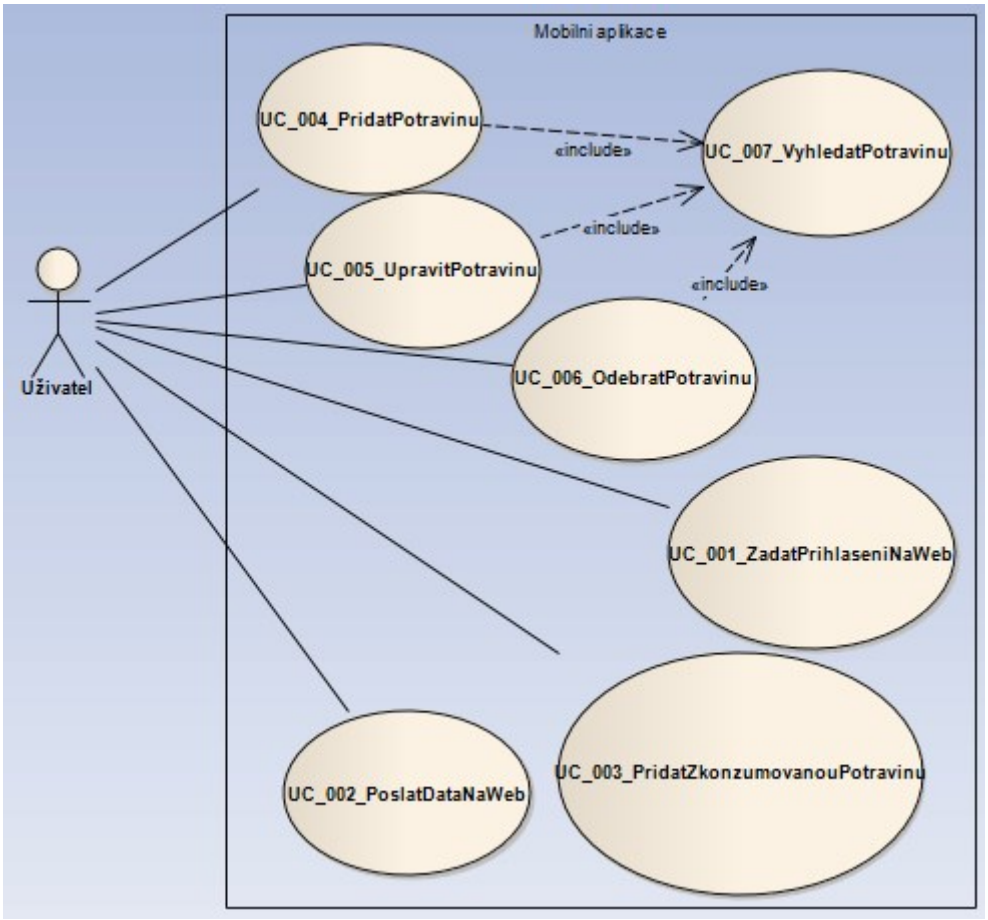
UC_006_PrijmoutDataZMobilu: Systém přijme data ve formě XML z mobilního zařízení. Uloží je do adresáře.

UC_007_KontrolovatUcet: Případ užití je spuštěn systém při uložení přijatého XML souboru. Systém rozloží část XML souboru a získá z něj přihlašovací údaje. Systém porovná přihlašovací údaje s údaji v databázi. Pokud se údaje shodují, systém umožní další zpracování dat.

UC_008_UlozitData: Případ užití začíná shodou přihlašovacích údajů. Systém rozloží zbytek XML souboru, získá z něj data o denním příjmu a nově přidaných potravinách. Data uloží do databáze.

UC_009_ZaslatOdpoved: Mobilní aplikaci je vrácena odpověď ohledně výsledku přijetí dat.

6.2 Use Case diagram mobilní aplikace



Obrázek 5 – Diagram případů užití mobilní aplikace

Podle autora (Arlow, 2008, s. 112) je nejvhodnější modelovat případy užití, pokud má systém mnoho uživatelů, rozhraní a převládají funkční požadavky. Mobilní aplikace je už z podstaty zařízení pouze jednouživatelská. Ale díky většímu množství funkčních požadavků, je vhodné je pro přehlednost zaznamenat. Tudíž v tomto případě je zde aktérem uživatel, majitel mobilního zařízení. Dále jsou popsány jednotlivé případy užití, které spouští aktér.

UC_007_VyhledatPotravinu: Uživatel hledá údaje o potravině. Vybírá ze seznamu kategorií a podkategorií. Systém zobrazí konkrétní seznam potravin.

UC_004_PridatPotravinu: Případ užití zahrnuje UC_007_VyhledatPotravinu. Začíná, když uživatel zvolí „Přidat potravinu“. Systém zobrazí formulář pro zadání nové potraviny. Po kliknutí na „Uložit“, systém zkontroluje správnost zadaných hodnot ve formuláři. Dokud jsou hodnoty špatné, zobrazuje chybové hlášení, jinak novou potravinu uloží.

UC_005_UpravitPotravinu: Případ užití zahrnuje UC_007_VyhledatPotravinu. Je vyvolán, když uživatel zvolí v nabídce „Upravit“. Systém povolí upravit pouze potraviny přidané uživatelem.

UC_006_OdebratPotravinu: Uživatel odebere nesystémovou potravinu. Případ užití zahrnuje UC_007_VyhledatPotravinu. Začíná vybráním „Smazat“. Systém zobrazí dialogové okno, kde se ujistí, zda uživatel chce opravdu potravinu smazat. Pokud to uživatel potvrdí, systém smaže potravinu.

UC_001_ZadatPrihlaseniNaWeb: Případ užití je zahájen potvrzením položky v hlavním menu. Pokud již nějaké přihlašovací údaje existují, systém je zobrazí ve formuláři, v opačném případě se načte formulář prázdný. Uživatel zadá nebo upraví své přihlašovací údaje. Systém je uloží.

UC_003_PridatZkonzumovanouPotravinu: Uživatel přidá potravinu, kterou zkonzumoval. Případ užití nastává kliknutím na tlačítko „Přidat“. Uživatel vyplní formulář. Systém uloží zkonzumovanou potravinu.

UC_002_PoslatDataNaWeb: Posledním důležitým případem užití je poslání dat na web. Přenos není povolen, pokud uživatel nemá vyplněny přihlašovací údaje, jinak jsou data odeslána do webové aplikace. Systém zobrazí výsledek operace.

7 Analýza a návrh aplikace

Ještě před samotnou implementací aplikace bylo třeba nejdříve vytvořit diagramy tříd a navrhnout databázový model. Výsledné ukázky diagramů jsou zobrazeny v této kapitole.

7.1 Diagramy tříd

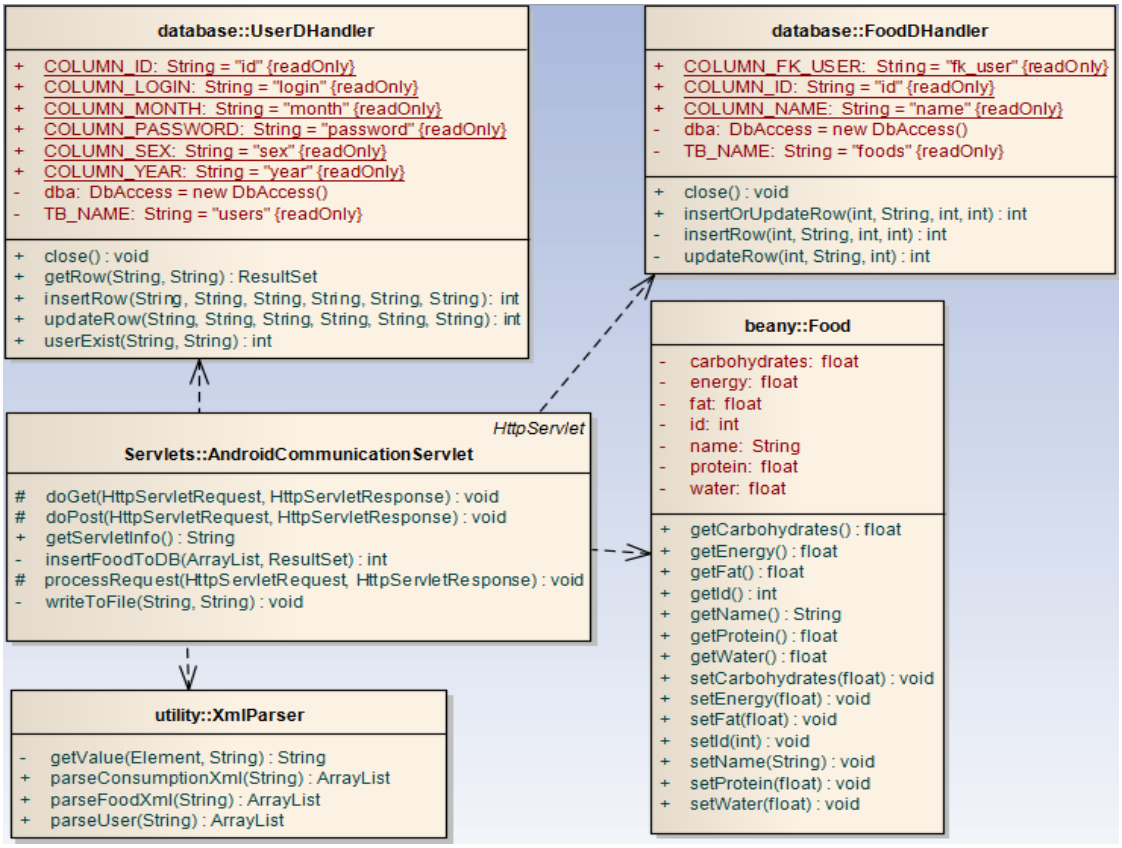
Diagramy tříd je třeba vytvořit ještě před implementací. Určují, jakou má aplikace strukturu, jaké se v ní využívají datové struktury a jaké jsou mezi nimi vztahy.

7.1.1 Diagramy tříd webové aplikace

Na úvod je třeba poznamenat, že webová aplikace je charakteristická tím, že dodržuje architekturu MVC. View zde představují JSP stránky, která zasílají data do controlleru neboli Servletu. Servlet volá pomocné třídy, většinou se jedná o Model část.

A vzhledem k tomu, že JSP stránka nepředstavuje žádnou třídu a Servlet většinou znovu zobrazí JSP stránku, tak se zde diagram tříd nedá modelovat jako celek, ale jako pouze část zachycující komunikaci mezi Servletem a pomocnými třídami.

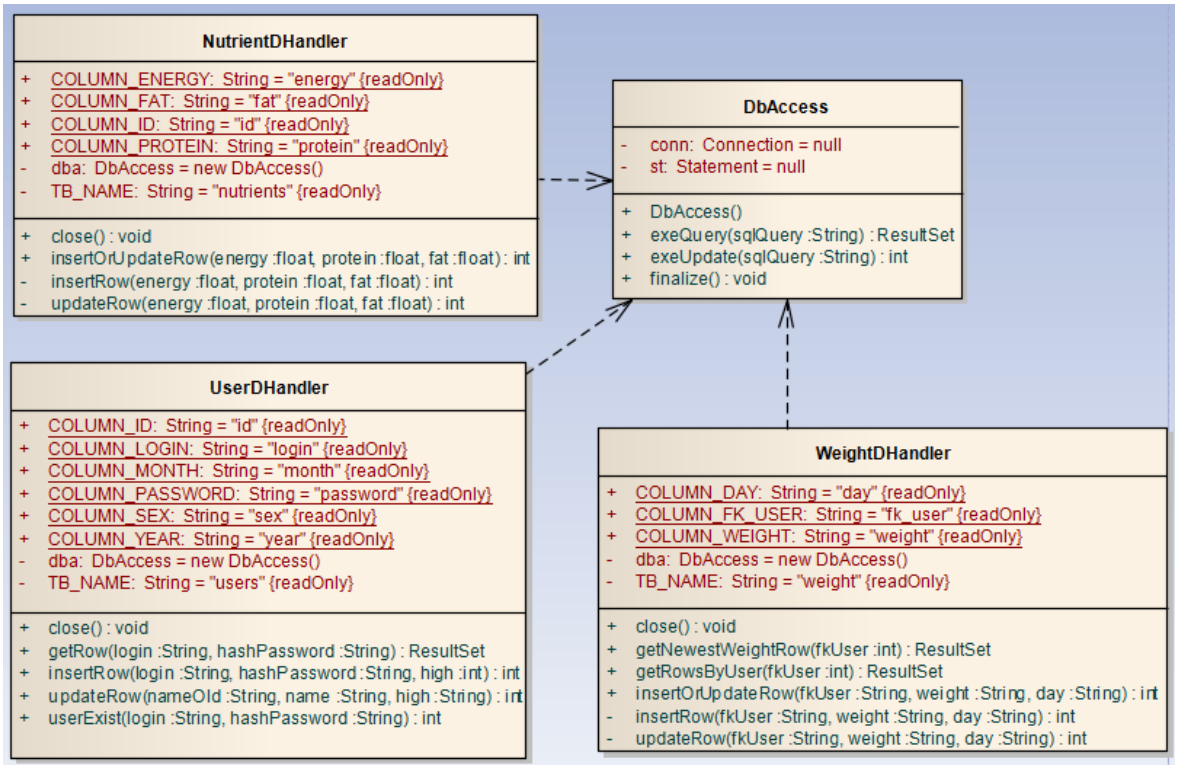
Na následujícím obr. 6 je příklad modelování takovéto časti. Servlet zde představuje třída `AndroidCommunicationServlet`, která je aktivována zasláním požadavku z mobilní aplikace. Servlet pomocí třídy `XmlParser` data z požadavku rozloží a uloží je do tříd s příponou `DHandler` které přistupují ke konkrétním tabulkám v databázi.



Obrázek 6 – Diagram tříd Servletu a pomocných tříd

Další ukázkou diagramu tříd jsou třídy, které získávají a ukládají data do databáze. Jejich název odpovídá názvům tabulkám, ke kterým zpravidla přistupují. Všechny tyto třídy závisí na třídě `DbAccess`. Funkcí třídy `DbAccess` je registrace Oracle ovladače, založení připojení k databázi v konstruktoru. A dále provádí vládání dat nebo jejich úpravu v tabulkách pomocí funkcí `exeQuery(String)` a `exeUpdate(String)`.

Aby se diagram vešel na stránku, jsou pro jednoduchost na následujícím obr. 7 a taktéž i na předchozím obr. 6 některé třídy upraveny. Konkrétněji byly odebrány nebo upraveny některé atributy a metody.

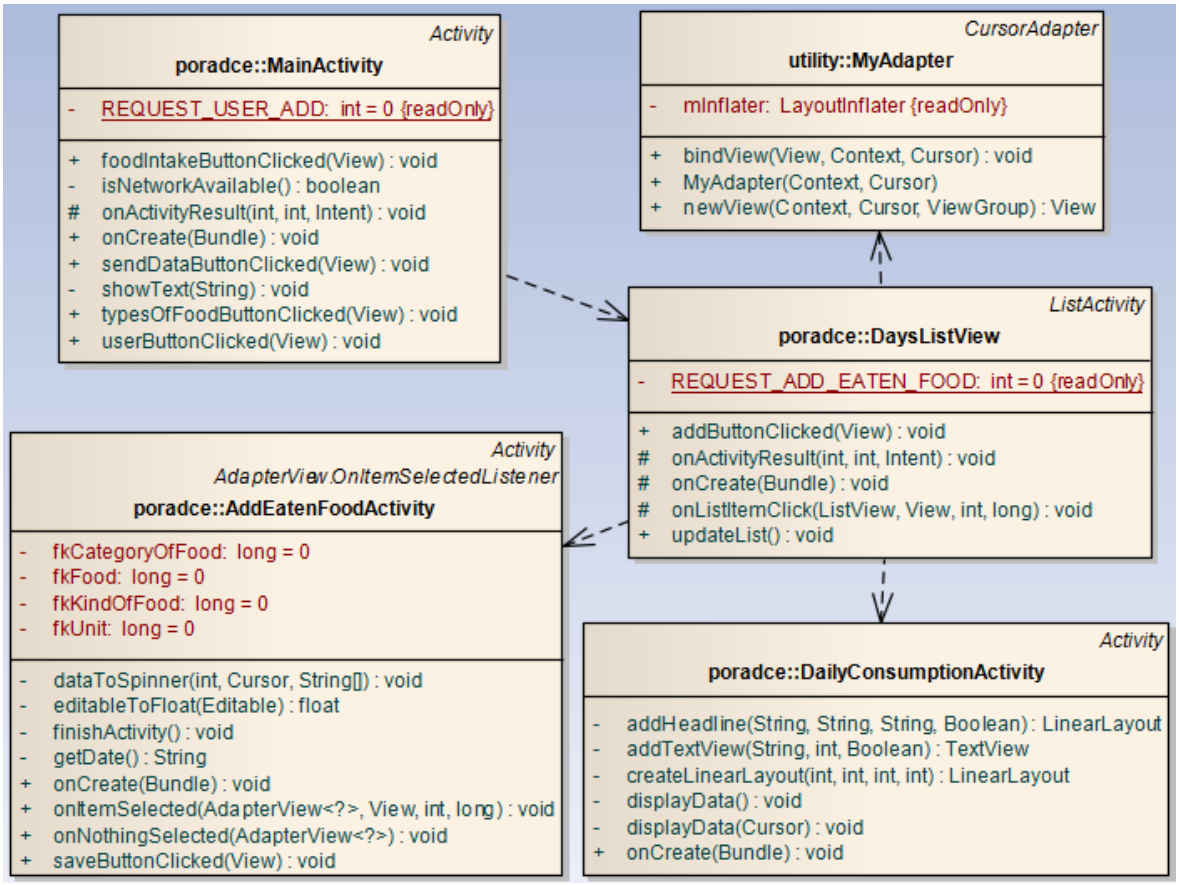


Obrázek 7 – Diagram tříd přistupujících k databázi

7.1.2 Diagramy tříd mobilní aplikace

Diagram tříd mobilní aplikace je rozdělen na menší části, protože jako celek by se na stránku nevešel a navíc lze tak lépe popsat jednotlivé třídy. Pro přehlednost diagramu jsou také vynechány ve třídách některé nepodstatné atributy, funkce a vazby na třídy, které mají název podle tabulek v databázi a získávají z nich data.

Hlavní třídou, od které se vše odvíjí, je `MainActivity.java`. A tato třída je první, kterou uživatel, při spuštění aplikace, uvidí. V metodě `onCreate()` načte uživatelské rozhraní z XML a inicializuje databázi. Všeobecně aktivity slouží pro tvorbu uživatelského rozhraní. Jejich struktura je rozebrána v kapitole o implementační části na konkrétním příkladu.



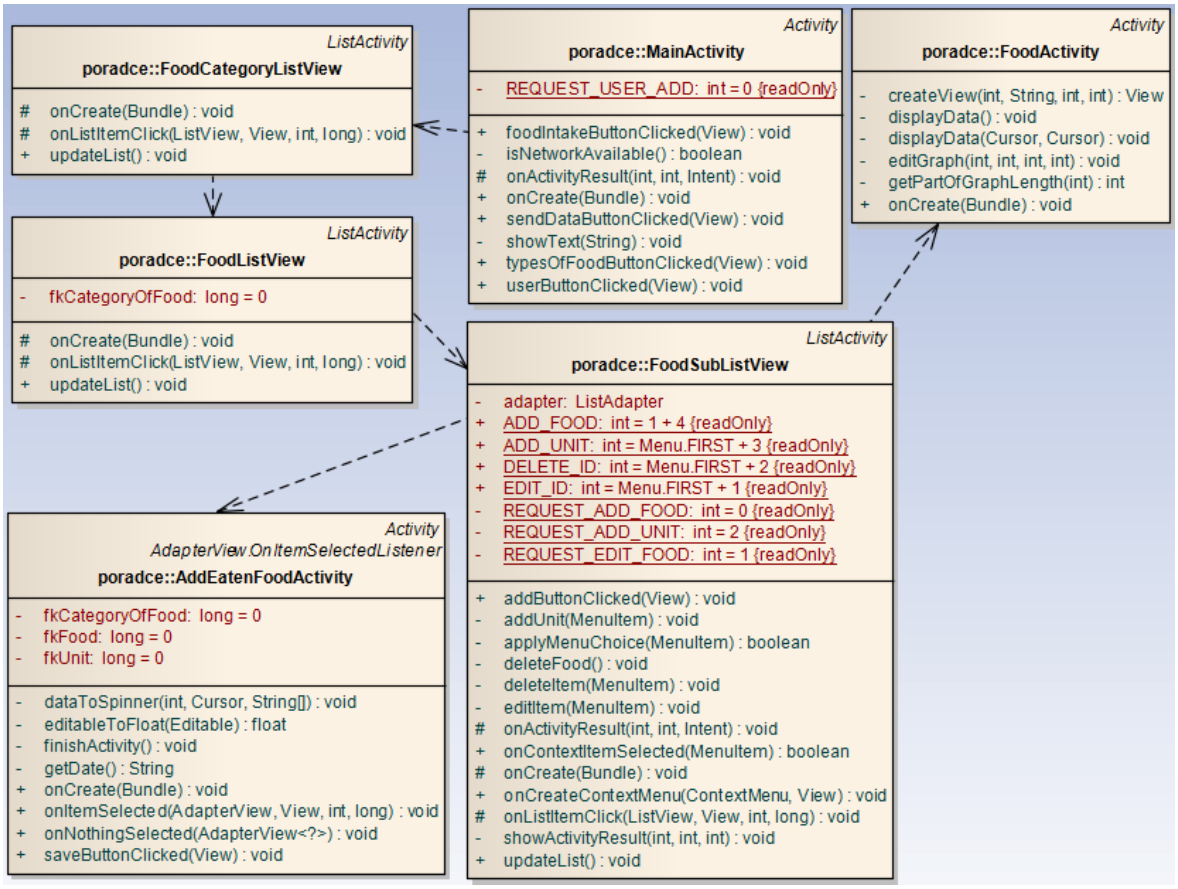
Obrázek 8 – Diagram tříd denní spotřeba

Část diagramu tříd na obr. 8 se zabývá zadáváním a zobrazováním zkonzumovaných jídel uživatele. Poté, co uživatel klikne na tlačítko „Kalendář jídel“, se zavolá metoda `foodIntakeButtonClicked()` z třídy `MainAcitivity`, která si přes záměr, neboli Intent, vyžádá spuštění třídy `DaysListView`.

Třída `DaysListView` má na starosti přes metodu `onCreate()` vypsání seznamu dnů, kdy uživatel zadával zkonzumované potraviny. Obvykle by k tomu využívala třídu `CursorAdapter`, ale jelikož je třeba dny získané z databáze upravit ze tvaru yyyy-MM-dd na tvar dd.MM.yyyy, třída `CursorAdapter` musela být rozšířena třídou `MyAdapter`.

Klikne-li uživatel na některý ze dnů přes metodu `onListItemClic()` třídy `DaysListView`, vyžádá se spuštění aktivity `DailyConsumptionActivity`. Jejím úkolem je zobrazit v tabulce seznam potravin, které uživatel v daný den zkonzumoval, společně s jejich množstvím a jednotkou. Metoda `addHeadline()` zobrazí hlavičku tabulky a řádky jsou přidány metodou `addTextView`.

Klikne-li uživatel ve třídě `DaysListView` na tlačítko „Přidání zkonzumované potraviny“, bude metodou `addButtonClicked()` přesměrován na aktivitu `AddEatenFoodActivity`. Tato aktivita zobrazuje formulář pro přidání zkonzumované potraviny. Ve formuláři se vybírá potravina pomocí Spinner prvků, což je výčet kategorií a podkategorií potravin a na to slouží metody `dataToSpinner()`, `onItemSelected()`, `onNothingSelected()`.



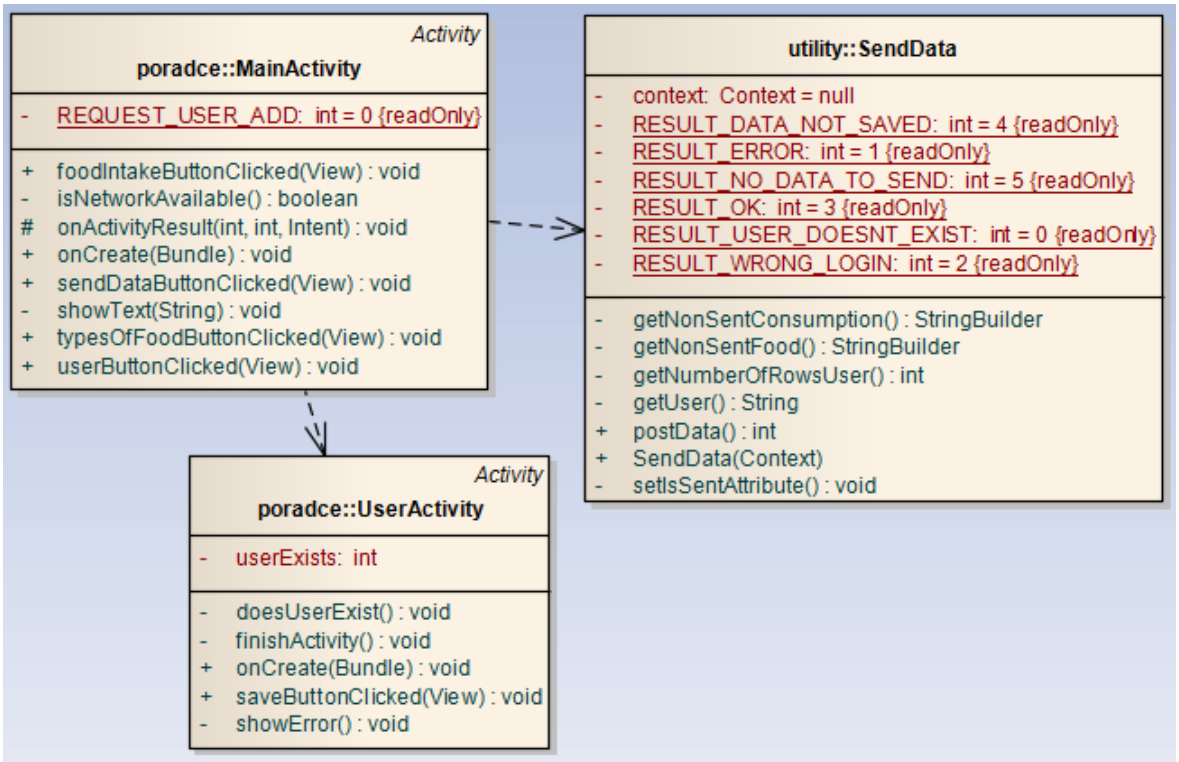
Obrázek 9 – Diagram tříd prohlížení a přidávání potravin

Zvolením tlačítka „Prohlížení potravin“ v `MainAktivity`, se uživatel přes metodu `typesOfFoodClicked()` dostane na aktivitu `FoodCategoryListView`. Zde se metodou `onCreate()` zobrazí seznam s kategoriemi jídel, např.: sladkosti. Poté, co si uživatel vybere některou z kategorií jídel, zavolá se metoda `onListItemClick()`, která vyžádá spuštění třídy `FoodListView` a ta zobrazí seznam podkategorií jídel.

Třída `FoodSubListView` vypisuje seznam konkrétních potravin. Podržením prstu na konkrétní potravině se zavolá metoda `onCreateContextMenu()`, která zobrazí menu s editací, mazáním potraviny a přidáním jednotky. Metoda `onContextItemSelected()` zachycuje, která položka v menu byla vybrána a volá podle toho metody editItem(), `deleteItem()` a `addUnit()`. Metoda `onListItemClick()` vyvolá aktivitu `FoodActivity` a metodou `addButtonClicked()` se uživatel dostane na třídu `AddEatenFoodAcitvity`. Výsledek těchto aktivit je zachycován v metodě `onActivityResult()`.

Přidávání nebo upravování jídel má na starosti třída `AddEatenFoodActivity`. Metodou `onCreate()` zobrazí formulář s prvky EditText a Spinner. Do Spinneru se nahrávají jednotky, které lze k potravině přidat metodou `dataToSpinner()`. Po kliknutí na tlačítko „Uložit“, metoda `saveButtonClicked()` zavolá metodu `finishActivity()`, která vrátí výsledek operace třídě `FoodSubListView`.

Třída `FoodActivity` zobrazuje konkrétní potravinu s grafem, představující poměr živin v této potravině metodami `getPartOfGraph()` a `createView()`.



Obrázek 10 – Diagram tříd zaslání dat

Poslední část diagramu tříd je na obr. 10. Skládá se ze tříd, které se starají o odesílání dat z mobilního zařízení na webovou aplikaci.

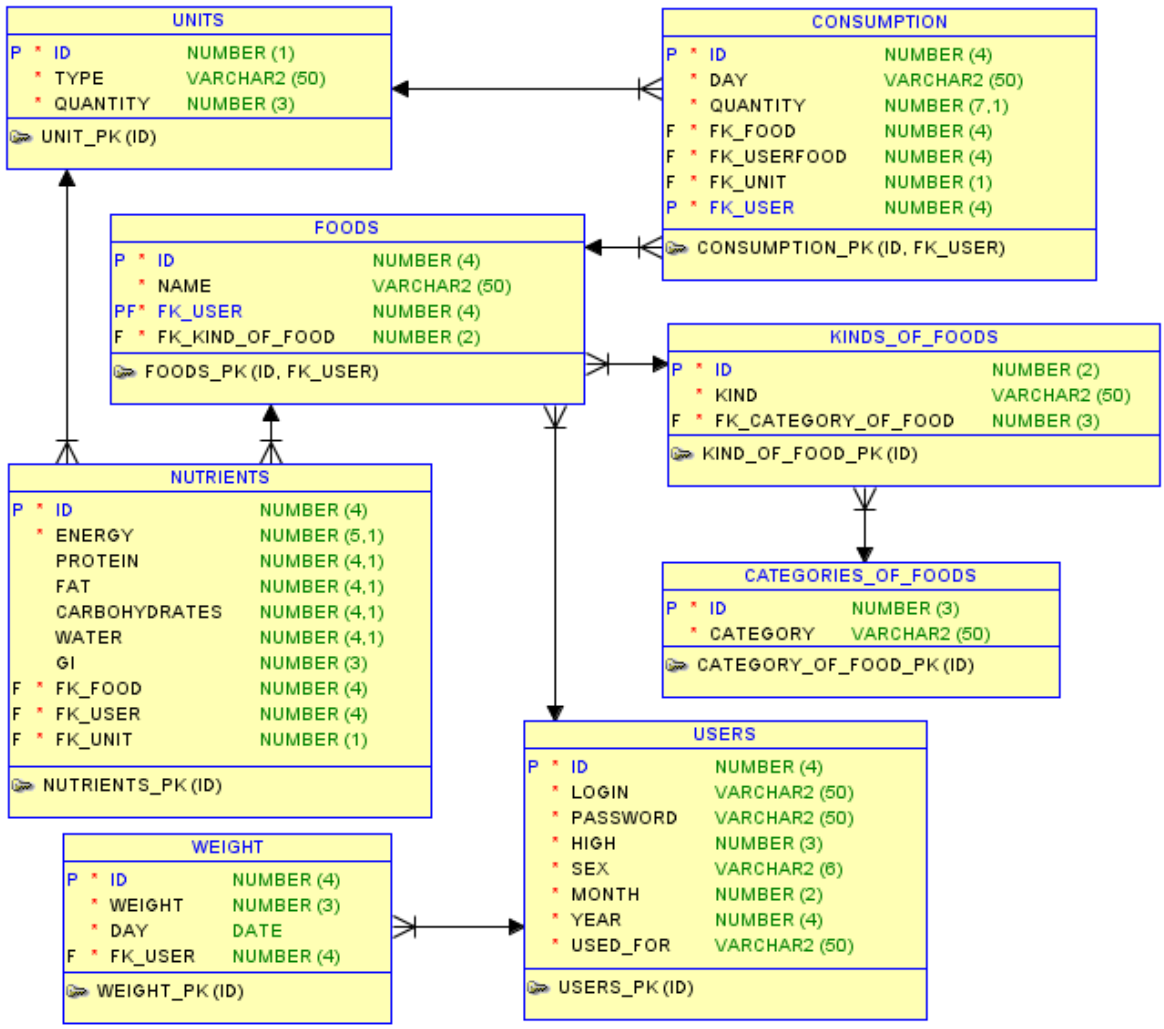
Konkrétně třída `UserActivity` načte v metodě `onCreate()` formulář pro zadání přihlašovacích údajů na web. Metoda `doesUserExist()` zajišťuje, aby pokud již uživatel data jednou zadal, znovu se načetly a mohl je tak editovat. Metodou `saveButtonClicked()` se data uloží a výsledek aktivity se zašle metodou `finishActivity()` zpět třídě `MainActvity`.

Třída `SendData` odesílá data do webové aplikace. Nejprve sestaví XML soubor z metod `getNonSentConsumption()`, `getNonSentFood()` a `getUser()`. Poté pošle tento soubor metodou `postData()`. V ní také přijímá zpět příznaky, jak operace s přijetím dat dopadla. Hodnoty příznaků jsou definovány jako konstanty ve statických atributech, např.: `RESULT_OK = 3`. Pokud vrácený příznak operace představuje, že vše dopadlo v pořádku, metodou `setIsSentAttribute()` se nastaví v tabulkách u poslaných řádků `is_sent` atribut na jedničku.

7.2 Návrh databázového modelu

Na základě diagramů tříd lze určit, jaké aplikace bude využívat data. Podle toho je třeba navrhnout tabulky, stanovit vztahy mezi nimi, určit primární klíče. Procesem návrhu databázového modelu se zabývá tato kapitola.

7.2.1 Relační model webové aplikace



Obrázek 11 – Relační model webové aplikace

Výše uvedený obrázek představuje relační model webové aplikace. Všechny vazby mezi tabulkami jsou povinné. To znázorňuje nepřerušovaná plná čára. Všechny vyjadřují kardinalitu vztahu 1:N šipkou, kde rozvětvení u paty šipky znamená N vztah.

Tabulky USERS a WEIGHT uchovávají údaje o uživateli. V aplikaci se předpokládá, že uživateli se může měnit časem váha. A jelikož je třeba vytvářet do minulosti statistiky optimálního množství živin na základě váhy, je třeba ji uchovávat v samostatné tabulce WEIGHT společně s atributem day. Tím se docílí získávání váhy pro konkrétní datum a proto atribut weight nemůže být součástí tabulky USERS.

Tabulka CATEGORIES_OF_FOODS představuje základní dělení potravin ve smyslu: mléčné výrobky, nápoje, sladkosti… Umožňuje tak uživateli se rychle zorientovat při hledání potraviny.

Ještě jemnější členění potravin je v tabulce KINDS_OF_FOODS. Pod konkrétními hodnotami si jde představit: zmrzlina, bonbony, zákusky. Cizím je tvořen primárním klíčem tabulky CATEGORIES_OF_FOODS.

Konkrétní potravinu např. Fidorku lze najít v tabulce FOODS. Jelikož potravinu může přidávat i uživatel, primární klíč je složen jak z id potraviny, tak z id uživatele, aby se odlišila od systémových potravin a potravin přidaných dalšími uživateli. Cizí klíč ukazuje na id tabulky KINDS_OF_FOODS.

Potraviny mohou být uchovávány v různých jednotkách. Od polévkových lžic, přes mililitry, až po gramy. Tyto data se nachází v tabulce UNITS.

Zkonzumované potraviny jsou uloženy v tabulce CONSUMPTION. Aby se dalo vyhledat, ke kterému uživateli záznam patří, primární klíč je složen z id a id uživatele fk_user, což je zároveň i cizí klíč. Pozor je třeba dát na cizí klíč fk_userfood. Ten společně s fk_food jednoznačně určuje potravinu, kterou uživatel zkonzumoval. Cizí klíč fk_unit představuje, pro jaké množství a v jaké jednotce byla potravina spotřebována, např. 5 kusů.

Poslední tabulkou je tabulka NUTRIENTS. Ukládá množství živin k jednotlivým potravinám. Na ty se odkazuje pomocí cizích klíčů fk_food a fk_user. Na první pohled jde vidět, že ne všechny atributy jsou povinné. Cizí klíč fk_unit udává živiny pro danou jednotku: 100g, 100ml apod. Ze začátku tvorby aplikace totiž ještě nebylo jasné, které položky ve formuláři bude muset uživatel povinně vyplnit.

7.2.2 Relační model mobilní aplikace

Následující obr. 12 představuje relační model mobilní aplikace. Na první pohled vypadá stejně jako model webové aplikace, avšak je tam pár zásadních rozdílů.

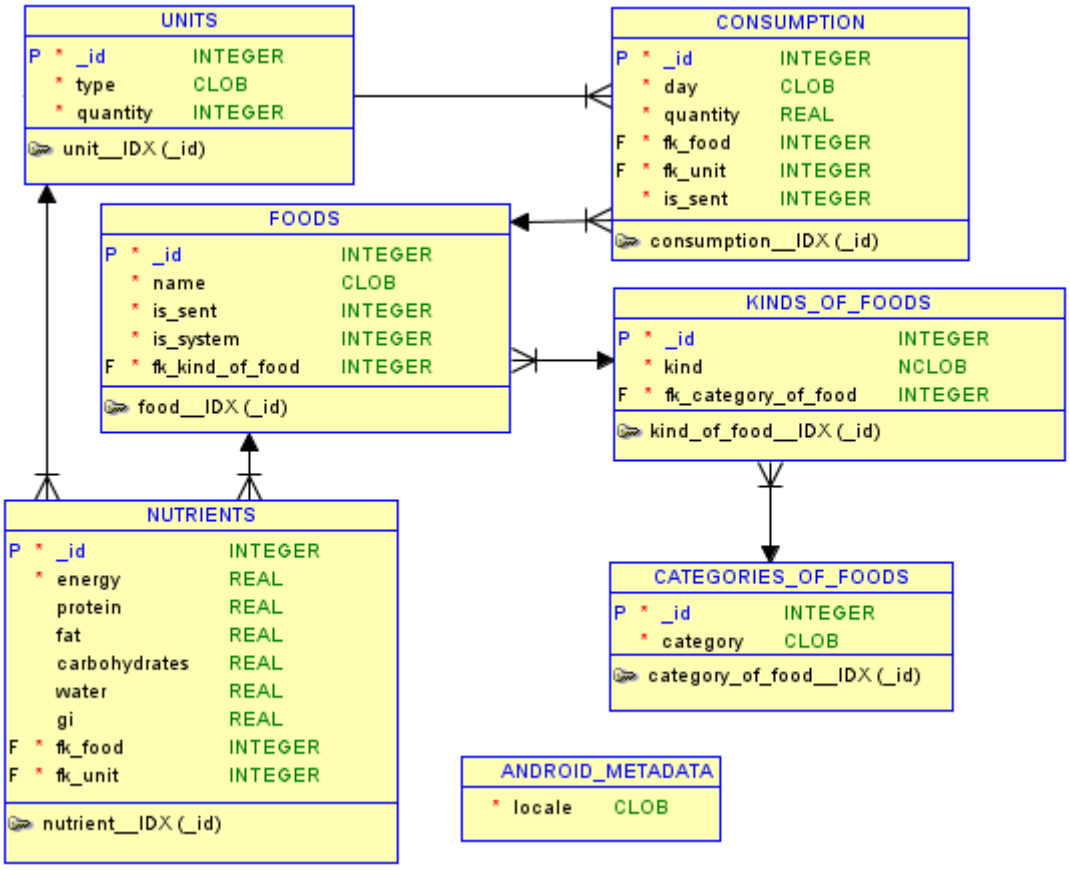
Protože aplikace je pouze pro jednoho uživatele, majitele mobilního zařízení a nevytváří žádné statistiky, ke kterým by potřebovala údaje o váze, prvním viditelným rozdílem je určitě odebrání tabulek WEIGHT a USERS.

Nicméně aby mobilní aplikace věděla, které nově vytvořené potraviny má poslat do webové aplikace, stále je třeba evidovat údaje, jestli se jedná o potravinu systémovou nebo potravinu vytvořenou uživatelem. A tak byl atribut fk_user nahrazen atributem is_system. A do tabulek FOODS a CONSUMPTION přidán atribut is_sent značící, jestli mobilní aplikace tyto data již odesílala a neposílaly se tak zbytečně znova.

Tabulka ANDROID_METADATA představuje specialitu Androidu. Je vyžadována k uchovávání meta informací o aplikaci.

Dále došlo ke zjednodušení datových typů atributů, které již nejsou tak přesně definované jako u relačního modelu webové aplikace. A to z toho důvodu, že model byl vytvářen

pro databázi SQLite, která má pouze pět základních datových typů. Navíc podle Murphyho (2011, s. 231) SQLite uvedení datových typů považuje pouze za doporučení. Do jakéhokoliv atributu je možné uložit data jakéhokoliv typu.



Obrázek 12 – Relační model mobilní aplikace

8 Popis implementace

V této kapitole jsou popsány nejdůležitější části tvorby obou aplikací. Nejprve je rozebrána implementace webové aplikace, poté mobilní aplikace a nakonec je vysvětleno, jak aplikace mezi sebou komunikují.

8.1 Implementace webové aplikace

Nejprve bylo třeba zahájit implementaci webové aplikace. Nejzajímavější části tvorby jsou uvedeny v tomto oddíle.

8.1.1 Implementace architektury MVC

Architekturou MVC v rámci JSP stránek se zabývá článek (Java Server Pages, 2013). Tato architektura zabraňuje, aby uživatel přistupoval přes webový prohlížeč přímo k webové stránce. Pouze Servlet rozhoduje o tom, která stránka se načte. Tato architektura se skládá ze tří komponent: model, view a controller. Vytvářejí se tak přehlednější a méně na sobě závislé stránky, které jsou, co se týče úprav, snadněji udržitelné.

Model

Model jsou třídy, které představují data, s nimiž aplikace pracuje. Nezávisí na JSP stránkách. Lze je tedy využít jak ve webové, tak i v desktopové nebo i v mobilní aplikaci. Obvykle se pro něj používají běžné třídy. `UserDHandler` může být ukázkou třídy, která reprezentuje komponentu model. Slouží k práci s daty v tabulce USERS. Obsahuje soukromý atribut uchovávající připojení k databázi a atributy s názvy sloupců v tabulce USERS, což jsou statické řetězové konstanty. Využívají se na příklad v dotazech nebo v instanci třídy `ResultSet`, kdy je třeba ve volané metodě zadat pro získání hodnoty název sloupce, tudíž díky těmto atributům se nezanáší do kódu chybné pojmenování.

```
private final String TB_NAME = "users";
public static final String COLUMN_ID = "id";
public static final String COLUMN_LOGIN = "login";
public static final String COLUMN_PASSWORD = "password";
public static final String COLUMN_HIGH = "high";
public static final String COLUMN_SEX = "sex";
private DbAccess dba = new DbAccess();
…
```

Pomocí metod třídy `UserDHandler` se pracuje s tabulkou USERS. Následující zdrojový kód představuje příklad takovéto metody. Nejdříve se pomocí zmíněných atributů sestaví dotaz. Dotaz se vykoná a vrátí se `ResultSet` objekt.

```
public ResultSet getRow(String login, String hashPassword) {
      ResultSet rs = null;
      String query = "SELECT *  FROM " + TB_NAME + " WHERE " +
          COLUMN_LOGIN + " = '" + login + "' AND " + COLUMN_PASSWORD +
          " = '" + hashPassword + "'";
      rs = dba.exeQuery(query);
      return rs;
}
```

Controller

Tato komponenta propojuje view s modelem. Jedná se o třídu reagující na požadavky od uživatele. Na jejich základě mění model nebo view. Controler je realizován Servletem a využívá pomocné třídy.

Zástupcem komponenty controller může být třída `EditUserServlet`. Stará se o úpravu dat uživatele. Uživatel vyplní v JSP stránce formulář a odešle ho. `EditUserServlet` zachytí přicházející HTTP požadavek. Zkontroluje data v požadavku, a pokud nevyhovují podmínkám, vrátí je zpět do formuláře ve view společně s chybovým hlášením.

```
String weight = (String) request.getParameter("weight");
if (Float.parseFloat(weight) < 1 || Float.parseFloat(weight) > 200) {
                    errorMessage.append("Váha má špatnou hodnotu.\n");
request.setAttribute("weight", weight);
request.setAttribute("error", errorMessage);
RequestDispatcher view = request.getRequestDispatcher("edit_user.jsp");
```

Další zdrojový kód ukazuje, jak `EditUserServlet` přistupuje k modelu a mění v něm data. Jedná se o editaci váhy uživatele. Nejdříve se získají data z jednoho modelu, neboli vytvoří se objekt typu `UserDHandler`. Získají se z něj informace o uživateli a na jejich základě controller zaktualizuje data v modelu `WeightDHandler`.

```
UserDHandler user1 = new UserDHandler();
ResultSet rs1 = user1.getRow(name, null);
rs1.next();
WeightDHandler wdh = new WeightDHandler();
…
wdh.insertOrUpdateRow(rs1.getString("id"), weight, ft.format(now));
```

Conroller také volí jaké view se zobrazí. Zde se kontroluje, zda uživatel zadal správné heslo a podle toho buď controller přesměruje požadavek na další JSP stránku nebo ho vrátí na původní stránku pro opětovnou úpravu přihlašovacích údajů.

```
if (userExists == 1) {
    RequestDispatcher view = request.getRequestDispatcher("test.jsp");
    view.forward(request, response);
} else {
    request.setAttribute("spravneZadaneHeslo", "false");
    RequestDispatcher view=request.getRequestDispatcher("edit_user.jsp");
    view.forward(request, response);
}
```

View

Controller předává komponentě view obsah z modelu a ta jej zobrazí uživateli. Takže view představuje uživatelské rozhraní, které je tvořeno s částí controller. Jako view se zde využívají JSP stránky.

Zvolenou ukázkou view komponenty ve webové aplikaci je edit_user.jsp. V tomto příkladu uživatel špatně vyplnil pole ve formuláři a vrací se mu ze Servletu-controlleru zpět. View data z požadavku přijme a zobrazí je.

```
<%
      String name = (String) request.getAttribute("name");
      String error = (String) request.getAttribute("error");
      …
%>
<form action="EditUserServlet" method="post">
      <p>E-mail (login):</p>
      <p><input type="text" name="name" value="<%= name%>"/></p>
      <p>Váha: </p>
      <p><input type="text" name="weight" value="<%= weight%>"/></p>
      …
      <input type="submit" name="register" value="Uložit" />
</form>
```

8.1.2 Implementace statistik

Pro tvorbu grafů a tabulek ve webové aplikaci byla vybrána technologie Google Chart Tools. Nabízí kolem patnácti typů různých grafů, které jdou snadno vytvářet a upravovat. Pro jejich tvorbu využívá pouze JavaScript a není tak třeba přidávat žádné dodatečné knihovny. Následující příklad demonstruje, jak se postupovalo při tvorbě sloupcového grafu, který zobrazuje poměr skutečně dodaných živin a jejich doporučenou denní dávku. Průvodcem při implementaci byla webová stránka (Google Chart Tools — Google Developers, 2012).

Nejprve je třeba nahrát tři knihovny: Google JSAPI API, Google Visualization a knihovnu pro graf. V ukázce lze vidět, že první element `<script>` nahrává JSAPI knihovnu. Přes druhý element `<script>` se nahrává Google Visualization a knihovny pro tabulku i sloupcový graf. Po zavolání funkce `google.load()` by měla v kódu následovat metoda `google.setOnLoadCallback()`, na kterou se odvolává, když je nahráno Visualization API.

```
<script type="text/javasript" src="http://www.google.com/jsapi"></script>
<script type="text/javascript">
    google.load('visualization', '1',{'packages':['table','corechart']});
    google.setOnLoadCallback(draw);
    …
</script>
```

Dále je třeba dodat grafům popisky a data. Data se musí zabalit do JavaScript třídy zvané `google.visualization.arrayToDataTable`. Každý řádek v této tabulce představuje skupinu sloupců, kde první řádek udává popisek sloupců a další řádky už vkládají konkrétní hodnoty pro danou živinu.

```
var data1 = google.visualization.arrayToDataTable([
      ['Živiny', 'Skutečné množství', 'Optimální množství'],
      ['Bílkoviny',  <%= sumProtein%>, <%= (proteinUp + proteinDow)/2%>],
      ['Tuky',  <%= sumFat%>, <%= (fatDow + fatUp) / 2%>],
      ['Sacharidy',<%= sumCarbohydrates%>,<%= (carboDown + carboUp)/2%>]
]);
```

Každý graf je možné také přizpůsobit různým nastavením. Jedná se o barvy, tloušťku, osy, titulek, výplň pozadí… V tomto grafu byl navíc přidán titulek a osy. Aby byly osy více viditelné, byla jim navíc přiřazena modrá barva.

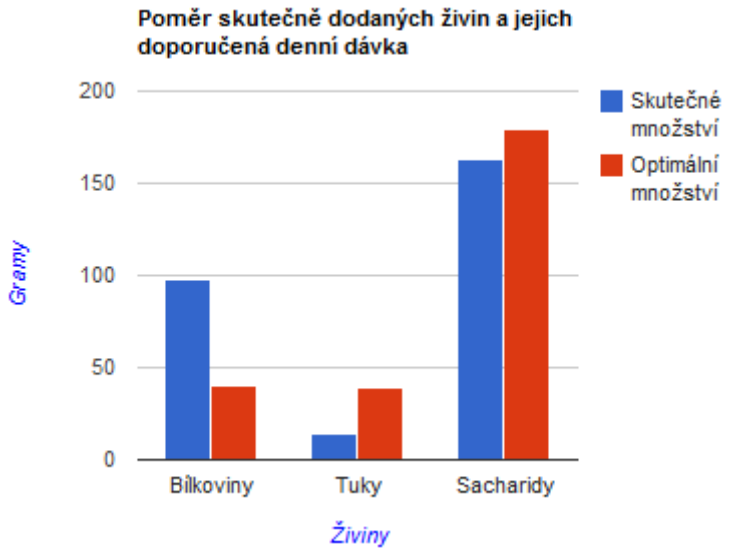
```
var options1 =
title:'Poměr skutečně dodaných živin a jejich doporučená denní dávka',
      hAxis: {title: 'Živiny', titleTextStyle: {color: 'blue'}},
      vAxis: {title: "Gramy", titleTextStyle: {color: 'blue'}},
      chartArea: {width: '55%'}
};
```

Sloupcový graf je založen na třídě `google.visualization.ColumnChart`. Konstruktor této třídy má jediný parametr: element, jehož prostřednictvím se bude graf vykreslovat. Instance grafu se naplní daty a definovaným nastavením.

```
var chart1 = new google.visualization.ColumnChart(
                                document.getElementById('chart_div1'));
chart1.draw(data1, options1);
```

Poté, co je graf kompletní, je možné ho na stránce vykreslit. Využívá se pro to obvykle element `<div>` s atributem id, kam se zadává na graf odkaz. Je dobré specifikovat velikost grafu. Načítání grafu někdy může pár sekund trvat. Pokud graf nemá ve stránce definován kontejner o určité velikosti, může docházet při vykreslování grafu ke skokům v návrhu stránky. Výsledný graf je znázorněn na obr. 14.

```
<div id="chart_div1" style="width: 400px; height: 300px; position:
relative;"></div>
```



Obrázek 13 – Graf vytvořený pomocí Google Chart Tools

8.1.3 Implementace čtení dat z XML souboru

Parsování neboli rozklad XML souboru je v aplikaci velmi důležitý. Získávají se tak data z mobilní aplikace, které je třeba uložit do databáze. Následující příklad ukazuje, jak se

z XML dokumentu získají data o uživateli z elementu <user>. Nejdříve se vytvoří objekt typu DocumentBuilderFactory, který umožní aplikaci získat parser. DocumentBuilder vytvoří XML dokument. Nakonec se vytvoří instance třídy Document pomocí metody parse(), která rozloží obsah XML dokumentu. Třída Document přistupuje k datům dokumentu. Vytvoří se takto rozhraní představující celý XML dokument.

```
DocumentBuilderFactory
      docBuilderFactory=DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
Document doc = docBuilder.parse(new File(path));
```

Dále je třeba získat například údaje o uživateli, které se nachází uvnitř elementu <user>. Metoda getElementsByTagName() vrátí seznam těchto elementů, tento seznam se v cyklu prochází a vybírají se data o konkrétním uživateli.

```
NodeList listUser = doc.getElementsByTagName("user");
for (int s = 0; s < listUser.getLength(); s++) {
     Node firstPersonNode = listUser.item(s);
     …
}
```

Při získávání dat je třeba ověřit, jestli objekt typu Node je elementem, protože může být také atributem, textovým uzlem aj. Třída Element rozšiřuje třídu Node a je určena speciálně pro uchovávání elementů. Z této instance třídy Element se získají elementy <login>. Metodou item() se získá první element a pomocí metody getNodeValue() se obdrží hodnota uvnitř tohoto elementu.

```
if (firstPersonNode.getNodeType() == Node.ELEMENT_NODE) {
      Element element = (Element) firstPersonNode;
      NodeList nodelist = element.getElementsByTagName("login");
      Element nodeElement = (Element) nodelist.item(0);
      NodeList textNode = nodeElement.getChildNodes();
      return ((Node) textNode.item(0)).getNodeValue();
}
```

Obdobně se pak získají data z elementu <password>. A celý proces se bude opakovat pro elementy <consumption> a <food>, které představují nově zkonzumované potraviny a nově přidané potraviny.

8.1.4 Implementace vzhledu aplikace

Vzhled aplikace je velice důležitý pro vytvoření dojmu, při prvním setkání uživatele s aplikací. Prvních několik sekund, po příchodu na stránku, je rozhodujících pro setrvání uživatele. Pokud se v této krátké době nedokáže uživatel na stránce zorientovat a nalézt nějaký záchytný bod, který ho zde udrží, tak z takovéto stránky odchází a již se sem s velkou pravděpodobností nevrátí.

Nejdůležitějším faktorem při návrhu uživatelského rozhraní, je intuitivnost a snadná použitelnost při práci s aplikací. Vše musí být přehledné, jednoduché a snadno dostupné.

Při návrhu rozhraní jsem nejdříve v grafickém editoru GIMP vytvořila statický vzhled aplikace. Snažila jsem se, aby byl lehký, vzdušný a hodil se k tématu, kterým se aplikace zabývá.

V dalším kroku jsem vytvořila šablonu za pomoci HTML a CSS. Základ stránky tvoří dva elementy DIV umístěné nad sebou, na obr. 13 jsou zvýrazněny červenými obdélníky. Horní element obsahuje pouze obrázek a název aplikace a má CSS třídu s názvem `hlavicka`. Dolní DIV element s třídou `telo` obsahuje další dva DIV elementy, které jsou zvýrazněny modrými obdélníky. Jeden je zarovnaný doleva, má třídu `menu`, a tvoří menu, druhý je zarovnaný doprava a obsahuje veškerý zobrazovaný obsah a má třídu `obsah`.

```
<!DOCTYPE html>
<html>
    <head>
      <title>Výživový poradce</title>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <link href="styl.css" media="screen" rel="stylesheet"
type="text/css" >
    </head>
    <body>
        <div class="hlavni">
          <div class="hlavicka">
            <div class="logo"></div>
            <h1>Výživový poradce</h1>
            <div class="login">odhlásit</div>
          </div>
            <div class="telo">
              <div class="menu">
                <ul>
                  <li><a href="">položka menu</a></li>
                </ul>
              </div>
              <div class="obsah">
              </div>
              <div class="cleaner"></div>
            </div>
        </div>
    </body>
</html>
```

Obrázek 14 – Rozložení div elementů

Vzhled stránek je vytvořen za pomoci technologie CSS. Zakulacené rohy jsou jednou z nových vlastností CSS verze 3. Zde je však omezení, že tato nová vlastnost nefunguje v prohlížeči Internet Explorer 8 a starších verzích, nicméně toto nemá žádný vliv na použitelnost stránek.

Správné zobrazení aplikace bylo otestováno v prohlížečích Mozilla Firefox, Google Chrome a Microsoft Internet Explorer 9.

8.2 Implementace mobilní aplikace

V tomto pododdíle je vysvětleno, jak se postupovalo při nastavování manifestu, co to jsou aktivity a jak se s nimi pracuje a na konci je popsáno, jak byl vytvářen vlastní adaptér.

8.2.1 Obsah manifestu

Základem aplikace pro Android je soubor AndroidManifest.xml a je dobře představen autorem (Murphy, 2011, s. 29-34). Generuje se automaticky při založení nového projektu. Nachází se v kořeni projektového adresáře. Jsou v něm přihlášeny aktivity, služby, nastavují se různá povolení, hlavní aktivita projektu, kterou uvidí uživatel při spuštění aplikace jako první.

Kořenem tohoto souboru je níže uvedený element. Do atributu `packege` se přiřazuje název balíčku, tím se definuje báze aplikace. Toto je užitečné, že pokaždé, když je třeba zapsat v manifestu nějakou třídu, postačí pak nahradit balíček tečkou. Takže místo specifikace třídy `com.example.poradce.MainActivity` se píše pouze `.MainActivity`.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.poradce"
    android:versionCode="1"
    android:versionName="1.0" >
</manifest>
```

Také je dobré nastavit element <uses-sdk>. Jedná se o potomka elementu <manifest> a měl by být nastaven jako první. Umožňuje spustit aplikaci na zařízení s definovanou verzí. Používá pro to dva atributy minSdkVersion a targetSdkVersion. První atribut udává opravdu tu nejnižší verzi SDK, kterou aplikace požaduje a je podmínkou, aby mohla být aplikace umístěna na web Android Market. Druhý atribut specifikuje úroveň API, pro kterou je aplikace navržena. Je důležité u něj nastavit co největší hodnotu. Při nízké hodnotě bude aplikace i na nových verzích Androidu používat starý vzhled. Při výběru hodnot těchto parametr byl důležitý obr. 2.

```
<uses-sdk
        android:minSdkVersion="4"
        android:targetSdkVersion="16" />
```

Dalším použitým elementem je <uses-permission>, který indikuje práva, které aplikace potřebuje pro správný běh. V tomto případě se jedná o povolení aplikace připojit se na internet a povolení přístupu k informacím o síti. Tato práva jsou využívána při přenosu dat z mobilní aplikace na webovou. Existuje další velké množství práv, které se mohou v aplikaci povolit. Bývá však pravidlem, že by jich mělo být co nejméně, neboť to může uživatele odradit od instalace.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>
```

Element <appliaction> definuje obsah aplikace. Jeho atribut icon určuje základní ikonu pro všechny komponenty aplikace. Popisek aplikace a její částí se nastavují atributem label. Atribut theme představuje odkaz na motiv, který je aplikován na celou aplikaci.

V manifestu musejí být přihlášeny všechny používané aktivity. Bez toho by nešly spustit. Registrace se dělá prostřednictvím elementu <aktivity>. Do atributu name se zapisuje název aktivity, label vloží popisek. Element <intent-filter> popisuje podmínky, za kterých se aktivita zobrazí. Hodnoty MAIN společně s LAUNCHER znamenají, že se tato aktivita zobrazí v seznamu aplikací.

```
<application
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
             <action android:name="android.intent.action.MAIN" />
             <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
      </activity>
      <activity android:name=".FoodListView" >
      </activity>
      <activity android:name=".UserActivity" >
      </activity>
</application>
```

8.2.2 Implementace třídy FoodSubListView

Jak uvádí autor (Murphy, 2011, s. 37-45), aktivity jsou speciální třídy v Androidu. Její implementace je v tomto pododdíle popsána na třídě `FoodSubListView`. Sice rozšiřuje třídu `ListActivity`, ale jelikož se jedná o podtřídu `Activity`, má všechny její vlastnosti a navíc umožňuje vytvářet jednodušeji výčty nabídek. Každá aktivita představuje jednu obrazovku a s tou mohou uživatelé pracovat.

Při spuštění aktivity se zavolá metoda `onCreate()`. V této metodě se vytvoří objekty, které aktivita bude obsahovat. Tyto objekty jsou definovány v XML souborech, zde konkrétně v souboru `food_sub_list.xml`.

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.food_sub_list);
      …
}
```

Zdrojový kód níže představuje zkrácenou verzi souboru `food_sub_list.xml`. Obsahuje definici tlačítka a výčtu nabídek, které se zobrazí v `FoodSubListView`. Pro následující výklad je nejdůležitější atribut `onClik`. Určuje, jak se bude jmenovat metoda, která se zavolá po klepnutí na tlačítko.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    … >
    <Button
        android:text="@string/food_sub_list_add"
        android:onClick="addButtonClicked"
        … />
    <ListView
        android:id="@android:id/list"
        android:layout_height="0dip"
        android:layout_weight="1"
        … />
     …
</LinearLayout>
```

Stejný název metody, který je uveden u atributu tlačítka v XML souboru, musí být definován i v třídě, která tento návrh přes `setContentView()` nahrává. Jinak kliknutí na tlačítko bude bez odezvy.

V této aplikaci je ale třeba, aby se po zmáčknutí tlačítka „Přidání nové potraviny“ nahrála nová aktivita `AddFoodActivity` na obrazovku. Jak to však sdělit programu? Tento problém popisuje (Murphy, 2011, s. 184-185). Zde nelze vytvořit pouze instanci třídy, jak se to dělá obvykle. Musí se pamatovat, že se jedná o aktivitu a ta má jiná pravidla. Nejdříve je třeba vytvořit záměr, instanci třídy `Intent`, která pomocí metody `startActivityForResult()` nebo `startActivity()` vyžádá, aby se aktivita `AddFoodActivity` spustila. Pokud je třeba aktivitě, která má být spuštěná, ještě předat data, využívá se k tomu instance třídy `Bundle`. Zde se například předává hodnota `fkKindOfFood`.

```
public void addButtonClicked(View button) {
      Bundle bundle = new Bundle();
      bundle.putLong("fkKindOfFood", fkKindOfFood);
      bundle.putInt("option", ADD_FOOD);
      Intent intent = new Intent(this, AddFoodActivity.class);
      intent.putExtras(bundle);
      startActivityForResult(intent, REQUEST_ADD_FOOD);
}
```

Rozdíl mezi metodami `startActivityForResult()` a `startActivity()` je, že u první metody se může implementovat metoda `onActivityResult()`, která se zavolá po ukončení spuštěné aktivity a může se zde zachytit například výsledek operace. Když se použije druhá metoda, aktivita po skončení spuštěné aktivity nic nezachycuje.

Zdrojový kód níže ukazuje implementaci metody `onActivityResult()`. Parametr `resultCode` obsahuje kód výsledku, který vrací dokončená aktivita. Parametr `requestCode` udává, s jakým jedinečným číslem byla aktivita spuštěná, takže se může určit, jaká aktivita dokončila svůj úkol. První podmínka prověřuje, jestli se vrácený `requestCode` rovná požadovanému. Pokud ano a kód výsledku operace nepředstavuje úspěšné dokončení ani návrat přes tlačítko zpět, zobrazí se zpráva s chybovým textem. Pokud však operace proběhla v pořádku, aktualizuje se výčet položek a zobrazí se zpráva o úspěšném přidání potraviny.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
      if (requestCode == REQUEST_ADD_FOOD || … ) {
      if (resultCode != RESULT_OK && resultCode != RESULT_CANCELED) {
            Toast.makeText(this, failureText, Toast.LENGTH_LONG).show();
      } else if (resultCode == RESULT_OK) {
            updateList();
            Toast.makeText(this, succesText, Toast.LENGTH_LONG).show();
      }} else if (requestCode == REQUEST_EDIT_FOOD) {
      …
      }
      super.onActivityResult(requestCode, resultCode, data);
}
```

8.2.3 Implementace výčtu nabídek s vlastním adaptérem

Výčet nabídek je v Androidu představen třídou `ListView`. Prostřednictvím adaptéru vkládá do řádků data. V základním nastavení lze do nich ukládat pouze text. Při tvorbě mobilní aplikace je však potřeba, aby v řádcích bylo něco jiného. Například ve třídě `FoodCategoryListView` je třeba uchovávat v řádcích obrázek s textem a v jiné třídě `DaysListView` je třeba měnit vkládané dny z formátu yyyy-MM-dd na dd.MM.yyyy.

Řešení je zde předvedeno na třídě `FoodCategoryListView`. Nejprve je třeba vytvořit vnořenou třídu, která rozšíří používaný adaptér. V tomto případě se jedná o adaptér `SimpleCursorAdapter`, který pracuje s daty uloženými v instanci třídy `Cursor`.

```
public class MyAdapter extends SimpleCursorAdapter{…}
```

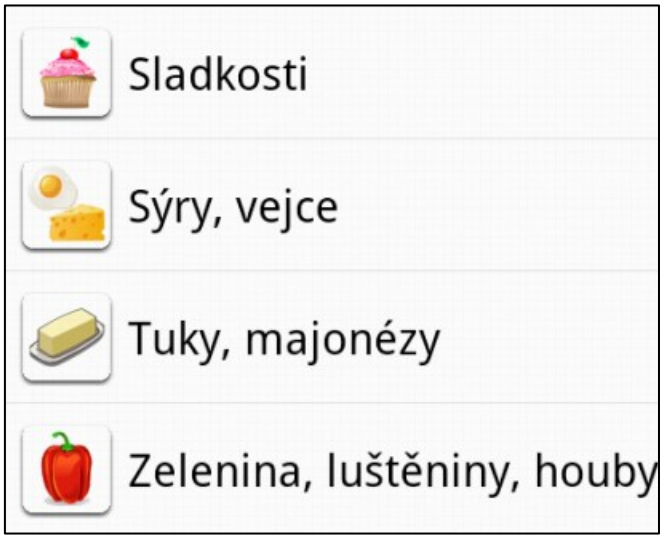
Dále se nahradí metoda `getView()`. Vstupními parametry jsou pozice v řádku, dodaný náhled a rodič tohoto náhledu. Navrací složený objekt typu `View`, představující řádek ve výčtu.

Pokud má náhled hodnotu null, je naplněn stromem objektů typu `View` z návrhu `R.layout.picture_spinner_item`. Instance kurzoru `Cursor` se posune na pozici určenou v parametru. Nakonec se zaplní řádek obrázkem a textem z kurzoru.

```
public View getView(int position, View convertView, ViewGroup parent) {
      if (convertView == null){
            convertView = View.inflate(context,
                                       R.layout.picture_spinner_item, null);
      }
      View row = convertView;
      c.moveToPosition(position);
      TextView label = (TextView) convertView
                              .findViewById(android.R.id.text1);
      ImageView icon = (ImageView) row.findViewById(R.id.icon);
      label.setText(this.c.getString(c
            .getColumnIndex(CategoryOfFood.COLUMN_CATEGORY)));
      icon.setImageResource(arr_images[position]);
      return row;
}
```

Použití této vnořené třídy `MyAdapter` ve třídě `FoodCategoryListView` je následovné. Vytvoří instance třídy `MyAdapter` s parametry specifikující, jaký sloupec má kurzor z množiny data vybírat, do jakého návrhu má tato data ukládat a nakonec se předá metodě `setListAdapter`. Celkový výsledek je znázorněn na obr. 15.

```
String[] from = { CategoryOfFood.COLUMN_CATEGORY };
int[] to = { android.R.id.text1 };
setListAdapter(new MyAdapter(FoodCategoryListView.this,
R.layout.picture_spinner_item, categoryOfFood.getRows(), from, to, 0));
```



Obrázek 15 – Výsledek implementace vlastního adaptéru

8.2.4 Implementace tříd pracujících s databází

Mobilní aplikace využívá databázi SQLite. Aby mohla přistupovat k jednotlivým tabulkám této databáze, byly vytvořeny speciální třídy, které se nazývají podle těchto tabulek a získávají z nich data nebo je do těchto tabulek ukládají. Obsahují instanci třídy `DatabaseHelper`, která rozšiřuje třídu `SQLiteOpenHelper`. Jedná se o pomocnou třídu starající se o otevírání databáze, pokud existuje. Pokud neexistuje, tak jí vytváří a pokud se vyskytne nová verze databáze, provede její aktualizaci. Implementují se zde metody `onCreate()`, `onUpgrade()` a `onOpen()`.

Následující metoda `onOpen()` je jednou z nejčastěji používaných. Databáze v aplikaci se totiž neotevře nebo nevytvoří, dokud se nezavolá jedna z metod `getReadableDatabase()` nebo `getWritableDatabase()`. A tato metoda je právě volána vždy při otevření databáze jednou z těchto dvou metod, po zavolání `onCreate()` nebo `onUpgrade()`.

```
public class DatabaseHelper extends SQLiteOpenHelper {
      …
      @Override
      public void onOpen(SQLiteDatabase db) {
            super.onOpen(db);
      }
}
```

Nejvhodnější ukázkou třídy, která přistupuje ke konkrétní tabulce v databázi, může být například třída `Unit`. Manipuluje s daty v tabulce UNITS, která představuje, v jakých jednotkách může být potravina uložena: gramy, lžičky, kusy. Atributy třídy obsahují název tabulky a názvy sloupců, pro jejich snadné využití v dotazech.

```
protected static final String TB_NAME = "unit";
public static final String COLUMN_ID = "_id";
public static final String COLUMN_TYPE = "type";
public static final String COLUMN_QUANTITY = "quantity";
public static final String[] columns = { COLUMN_ID, COLUMN_TYPE,
                  COLUMN_QUANTITY };
protected static final String ORDER_BY = COLUMN_TYPE + " ASC";
private SQLiteOpenHelper openHelper;
```

Následující kód představuje konstruktor třídy Unit a ukázku metody přistupující k tabulce. Konstruktor vytvoří instanci výše zmíněné třídy, která se stará o otevírání databáze. Poté lze volat metodu getRows(), která obdrží data z tabulky a vrátí je obsažené v objektu Cursor. Otevře spojení s databází zmíněnou metodou getReadableDatabase(), protože data se z ní budou pouze číst. Provede se dotaz metodou query(), kde parametrem je název tabulky, pole názvů sloupců, které se mají získat a nakonec způsob seřazení dat.

```
public Unit(Context ctx) {
      openHelper = new DatabaseHelper(ctx);
}
public Cursor getRows() {
      SQLiteDatabase db = openHelper.getReadableDatabase();
      Cursor cursor = db.query(TB_NAME, columns, null, null, null, null,
                        ORDER_BY);
      return cursor; }
```

8.3 Implementace komunikace mezi mobilní a webovou aplikací

Při implementaci připojení k internetu a zasláním dat z mobilní aplikace se vycházelo z článku (Connecting to the Network, 2013).

8.3.1 Zaslání dat z mobilní aplikace

Zaslání dat začíná tak, že uživatel vybere tlačítko „Poslání potravin na web“. Jak už bylo popsáno, nejdříve je třeba v manifestu nastavit povolení pro přístup na internet. Diagram tříd, který modeluje tuto situaci, je na obr. 6. Zasílání dat z mobilní aplikace má na starost třída `SendData`. Ale před tím, než jí `MainAcitvity` zašle zprávu, je třeba v metodě `isNetworkAvailable()` zkontrolovat, jestli je mobilní zařízení připojeno k internetu použitím metod `getActiveNetworkInfo()` a `isConnected()`.

```
private boolean isNetworkAvailable(){
    ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
    return (networkInfo != null && networkInfo.isConnected())
}
```

Pokud metoda `isNetworkAvailable()` vrátí hodnotu true, vytvoří se instance třídy `SendData` a zavolá se její metoda `postData()`. Vytvoří HTTP klient, který je třeba k vykonání HTTP požadavků. Dále se vytvoří post hlavička, která jednoznačně identifikuje zdroj, kam se mají data poslat. V tomto případě se jedná o adresu webové aplikace, která je zatím provozována na lokálním aplikačním serveru bez přístupu z internetu.

```
HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost(
            "http://10.0.0.32:8080/Poradce/AndroidCommunicationServlet");
```

V dalším kroku je třeba shromáždit data, která je třeba poslat. Jedná se o přihlašovací údaje do webové aplikace, uživatelem nově přidané potraviny a jeho nově zaznamenaný denní příjem potravy. Jako jedna z možných struktur, kam se data dají seskupit, se zde nabízí XML struktura. Pro uložení dat v XML struktuře se využívají metody `getUser()`, `getNonSentFood()` a `getNonSentConsumption()`. Níže je vložena část kódu z metody `getNonSentFood()` a pod ní příklad dat tvořící XML strukturu.

```
StringBuilder foodData = new StringBuilder();
Food food = new Food(context);
Cursor foodCursor = food.getRowsByIsSent(0);
int rowsCount = foodCursor.getCount();
if (rowsCount > 0) {
      foodData.append("<food>");
      foodCursor.moveToFirst();
      while (!foodCursor.isAfterLast()) {
            foodData.append("<item>");
            foodData.append("<name>");
            foodData.append(foodCursor.getString(foodCursor
                        .getColumnIndex(Food.COLUMN_NAME)));
      }
      foodData.append("</food>");
}
```

```
<poradce>
      <user>
            <login>karel.vomacka@seznam.cz</login>
            <password> 955db0b81ef1989b4a4dfeae8061a9a6 </password>
      </user>
      <consumption>
            <item>
                  <day>2013-04-21</day>
                  <quantity>1000.0</quantity>
                  …
            </item>
      </consumption>
      <food><item> … </item></food>
</poradce>
```

Nakonec je třeba takto získaná data poslat do webové aplikace. URL se nastaví UTF-8 kódování pro POST parametry. Instance HTTP klienta zavolá metodu `execute()` a ta vykoná HTTP post požadavek.

```
httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs,"UTF-8"));
HttpResponse response = httpclient.execute(httppost);
```

8.3.2 Příjem a zpracování dat ve webové aplikaci

Mobilní aplikace zasílá data na Servlet `AndroidCommunicationServlet`. Ten je přijme přes požadavek v POST parametrech. Data následně uloží do XML souboru. Funkce `parseUser()` třídy `XmlParser` nejprve získá přihlašovací údaje uživatele z XML souboru. Poté se zkontroluje, jestli tyto údaje existují v databázi. Pokud nebyla nalezena shoda, odešle se mobilní aplikaci odpověď „userDoesntExist".

```
String xmlUser = (String) request.getParameter("XML");
String path = new File("food.xml").getAbsolutePath();
writeToFile(xmlUser, path);
XmlParser parser = new XmlParser();
ArrayList userAL = parser.parseUser(path);
UserDHandler user = new UserDHandler();
int isUser = user.userExist((String) userAL.get(0), passwordHash);
if (isUser == 1) {
      …
} else if (isUser == 0) {
      out.print("userDoesntExist");
}
```

Pokud však údaje o uživateli byly úspěšně nalezeny, rozkládá se XML soubor dál na jednotlivé nově přidané potraviny a denní spotřebu jídel. Takto získané hodnoty se ukládají do databáze, konkrétně do tabulek FOODS a CONSUMTION. Posledním krokem je odeslání odpovědi mobilní aplikaci o výsledku operací. Je třeba dát pozor na stav, kdy se data z nějakého důvodu do databáze neuložila, může se například jednat o špatně zadané hodnoty. Webová aplikace pak musí signalizovat mobilní, že nastala chyba. Tím se zabrání tomu, aby mobilní aplikace označila v databázi data jako odeslaná, ale přitom ve webové aplikaci by tato data chyběla a došlo by tak ke ztrátě integrity. Následující kód ukazuje řešení tohoto problému:

```
ArrayList foodAL = parser.parseFoodXml(path);
ArrayList consumtionAL = parser.parseConsumptionXml(path);
ResultSet userSet = user.getRow((String) userAL.get(0), passwordHash);
userSet.next();
int resultFood = insertFoodToDB(foodAL, userSet);
int resultCons = insertConsToDB(consumtionAL, userSet);
if (resultFood==0 || resultCons==0 || resultFood == 1 || resultCons == 1)
{
      out.print("dataWasSaved");
} else
{
  out.print("dataWasntSaved");
}
```

8.3.3 Zpracování výsledku operace v mobilní aplikaci

Webová aplikace tedy mobilní aplikaci, konkrétně třídě `SendData`, zasílá odpověď s výsledkem operace. Ta je uložena v instanci třídy `HttpResponse`. Pro získání obsahu odpovědi, je třeba `InputStream` převést na cílový datový typ, v toto případě na `BufferReader` a přečíst z výstupu řádek s výsledkem. Pokud operace dopadla úspěšně, odeslaným datům se změní v databázi příznak is_sent z nuly na jedničku.

```
HttpResponse response = httpclient.execute(httppost);
BufferedReader br = new BufferedReader(new InputStreamReader(
                        response.getEntity().getContent()));
String result = br.readLine();
if (result == null){
      return RESULT_NOT_CONNECTED;
} esle if (result.equals("userDoesnExist")) {
      return RESULT_WRONG_LOGIN;
} else if (result.equals("dataWasntSaved")) {
      return RESULT_DATA_NOT_SAVED;
} else if (result.equals("dataWasSaved")) {
      setIsSentAttribute();
      return RESULT_OK;
} else {
      return RESULT_ERROR;
}
```

Závěr

Cílem diplomové práce bylo vytvořit webovou a mobilní aplikaci pro výživové poradenství. Mobilní aplikace měla ukládat zkonzumované potraviny, ale také umožňovat vytvářet nové potraviny a posílat je do webové aplikace, kde se data zpracují a zobrazí výživové statistiky ve formě grafů. Webová aplikace měla umožňovat přihlášení více uživatelů zároveň. Grafické rozhraní mělo být pro uživatele intuitivní, přehledné a srozumitelné. Pro tvorbu těchto aplikací byl zvolen jazyk Java a JSP. Pro ukládání dat slouží databáze SQLite, Oracle 10g Express Edition. Splnění některých cílů mi dělalo potíže, ale nakonec se mi podařilo tyto překážky překonat a obě aplikace jsou připraveny k používání. Webová aplikace je zatím provozována na lokálním aplikačním serveru bez přístupu z internetu, ale jen do té doby, než se mi podaří najít kvalitní a cenově dostupný hosting.

Při tvorbě aplikace jsem ze začátku narážela na mnoho problémů. Nejdříve mi chyběla znalost správné výživy. Takže nejdříve jsem musela nastudovat, jaké je optimální množství bílkovin, tuků, sacharidů a příjem energie v závislosti na váze, výšce, věku a pohlaví. Jak toto množství ovlivňují různá onemocnění. O této problematice je naštěstí napsáno mnoho knižních i elektronických publikací. Další překážkou bylo, že jsem ještě neměla žádné zkušenosti s tvorbou aplikací pro Android. Zjistila jsem, že tvorba těchto aplikací sice není náročná, ale díky tomu, že to byla má první aplikace, tvorba postupovala velmi pomalu.

V implementační části bylo těžké zjistit, jakým způsobem se data budou přenášet z mobilní aplikace do webové. Toto jsem nakonec vyřešila zasláním dat pomocí protokolu HTTP metodou POST, které jsou uspořádány v XML struktuře.

Současné informace o bílkovinách, tucích, sacharidech, energii a glykemickém indexu patří mezi ty nejdůležitější údaje, podle kterých lze sledovat požadovanou dietu. Avšak ráda bych v budoucnu rozšířila aplikaci o další informace, díky kterým uživatel získá komplexnější přehled o své výživě. Jedná se například o dělení tuků. Tuků existuje několik druhů. Autorka (Laužanská, 2013) uvádí, že některé z nich způsobují zvýšené riziko vzniku srdečních chorob. Naproti tomu jiné usnadňují vstřebávání vitamínů rozpustných v tucích (A, D, E, K) a snižují hladinu cholesterolu v krvi. Proto bychom měli některým v jídelníčku dávat přednost. A líbilo by se mi v budoucnu toto dělení do aplikace přidat. Dále bych ráda rozšířila aplikaci o množství vlákniny, vitamínů, minerálů, glykemické náloži v potravinách. Údaje o vláknině by totiž ocenili lidé s vysokým cholesterolem, glykemickou nálož lidé s cukrovkou.

Díky této diplomové práci jsem mohla využít znalosti nabyté při magisterském studiu a naučila jsem se základům tvorby aplikace pro operační systém Android. Sice se v průběhu tvorby objevilo pár problémů, ale ty byly nakonec úspěšně překonány a obě aplikace jsou připraveny k použití. Věřím, že aplikace bude pro uživatele užitečná a že se v budoucnu bude moci rozšířit o další z navrhovaných informací.

Literatura

Android software development. 2013. In: *Wikipedia* [online]. [cit. 2013-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Android_software_development

ARLOW, Jim. 2008. *UML 2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press. ISBN 978-80-251-1503-9.

Civilizační choroba. 2013. In: *Wikipedie* [online]. [cit. 2013-05-08]. Dostupné z: http://cs.wikipedia.org/wiki/Civiliza%C4%8Dn%C3%AD_choroba

Comparison of mobile operating systems. 2013. In: *Wikipedia* [online]. [cit. 2013-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Comparison_of_mobile_operating_systems

Connecting to the Network. 2013. In: *Android Developers* [online]. [cit. 2013-05-08]. Dostupné z: http://developer.android.com/training/basics/network-ops/connecting.html#AsyncTask

Dashboards. 2013. In: *Android Developers* [online]. [cit. 2013-05-09]. Dostupné z: http://developer.android.com/about/dashboards/index.html

Dieta při kloubním onemocnění. 2012. In: *WikiSkripta* [online]. [cit. 2013-05-08]. Dostupné z: http://www.wikiskripta.eu/index.php/Dieta_p%C5%99i_kloubn%C3%ADm_onemocn%C4%9Bn%C3%AD

Enterprise JavaBeans. 2013. In: *FI WIKI* [online]. [cit. 2013-05-04]. Dostupné z: http://kore.fi.muni.cz/wiki/index.php/EJB

Google Chart Tools — Google Developers [online]. 2012 [cit. 2013-05-09]. Dostupné z: https://google-developers.appspot.com/chart/

IOS (Apple). 2013. In: *Wikipedie* [online]. [cit. 2013-05-05]. Dostupné z: http://cs.wikipedia.org/wiki/IOS_%28Apple%29#V.C3.BDvoj_pro_platformu_iOS

Java Platform, Micro Edition. 2013. In: *Wikipedia* [online]. [cit. 2013-05-05]. Dostupné z: https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition

Java Server Faces. 2013. In: *FI WIKI* [online]. [cit. 2013-05-04]. Dostupné z: http://kore.fi.muni.cz/wiki/index.php/JSF

Java Server Pages. 2013. In: *FI WIKI* [online]. [cit. 2013-05-04]. Dostupné z: http://kore.fi.muni.cz/wiki/index.php/JSP

LAUŽANSKÁ, Ilona. 2013. Tuky pod lupou. *Dieta*, březen 2013, roč. 10, č. 3, s. 60 – 61. ISSN: 1214-8784.

MÁLKOVÁ, Iva. 2005. *Hubneme s rozumem, zdravě a natrvalo*. Praha: Smart Press. ISBN 80-239-4112-7.

MEIER, Reto. 2010. *Professional Android 2 Application Development*. New Jersey: Wrox. ISBN 978-0-470-56552-0.

MÜLLEROVÁ, Dana. 2003. *Zdravá výživa a prevence civilizačních nemocí ve schématech*. Praha: TRITON. ISBN 80-7254-421-7.

MURPHY, Mark. 2011. *Android 2*. Brno: Computer Press. ISBN 978-80-251-3194-7.

NOURIE, Dana. 2006. Java Technologies for Web Applications. In: *Oracle* [online]. [cit. 2013-05-04]. Dostupné z: http://www.oracle.com/technetwork/articles/javase/webapps-1-138794.html

Oracle Documentation 10g Release 2. 2013. In: *Oracle* [online]. [cit. 2013-05-06]. Dostupné z: http://www.oracle.com/pls/xe102/homepage

PÁNEK, Jan. 2002. *Základy výživy*. Praha: Svoboda Servis, s. 55-56. ISBN 80-86320-23-5.

ŠTULC, Tomáš. 2001. Diagnostika a léčba hyperlipoproteinémií. In: *Medicina* [online]. [cit. 2013-05-08]. Dostupné z: http://www.medicina.cz/odborne/clanek.dss?s_id=1995

Top 8 Mobile Operating Systems form Mar to Arp 2013. 2013. In: *StatCounter Globlal Stats* [online]. [cit. 2013-05-05]. Dostupné z: http://gs.statcounter.com/#mobile_os-ww-monthly-201303-201304-bar

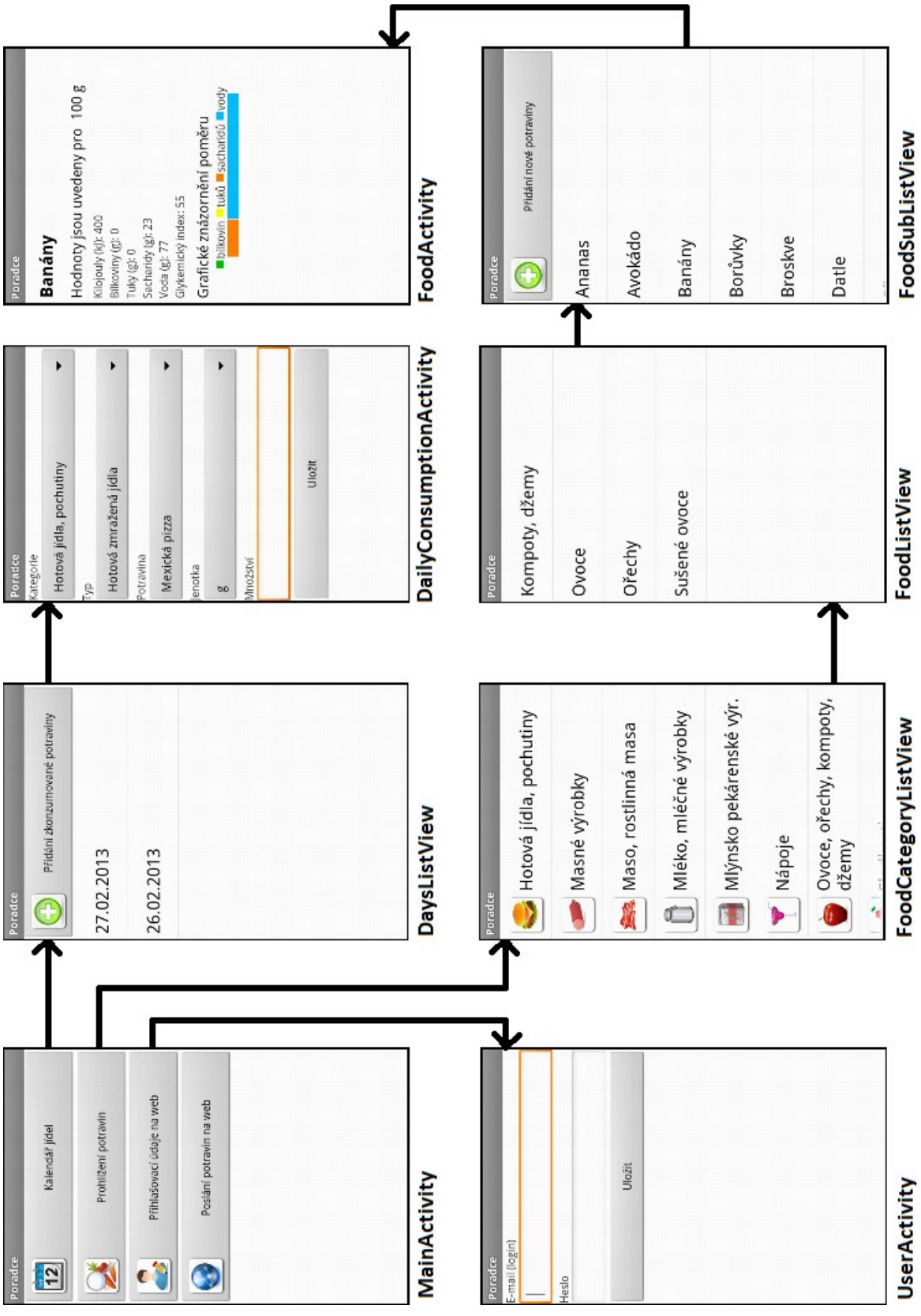
Tuky. 2012. In: *MTE* [online]. [cit. 2013-05-08]. Dostupné z: http://www.mte.cz/stravovani-tuky-lipidy.htm

UJBÁNYAI, Miroslav. 2012. *Programujeme pro Android*. Praha: Grada Publishing. ISBN 978-80-247-3995-3.

Vysoký cholesterol. 2012. In: *Vitalion* [online]. [cit. 2013-05-08]. Dostupné z: http://nemoci.vitalion.cz/vysoky-cholesterol/

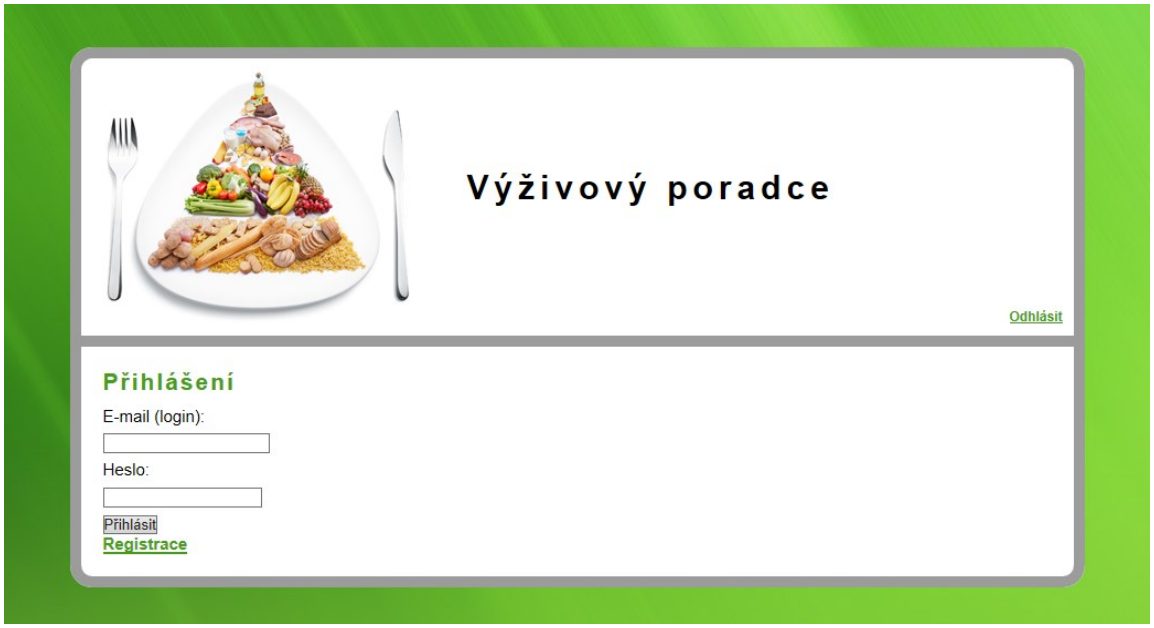
WELLS, Antonio. 2012. Latest versions of Android currently only accounts for 22 percent installed, most still on older Gingerbread. In: *AndroidTapp* [online]. [cit. 2013-05-09]. Dostupné z: http://www.androidtapp.com/latest-versions-of-android-currently-only-accounts-for-22-percent-installed-most-still-on-older-gingerbread/

Příloha A – Vzhled mobilní aplikace

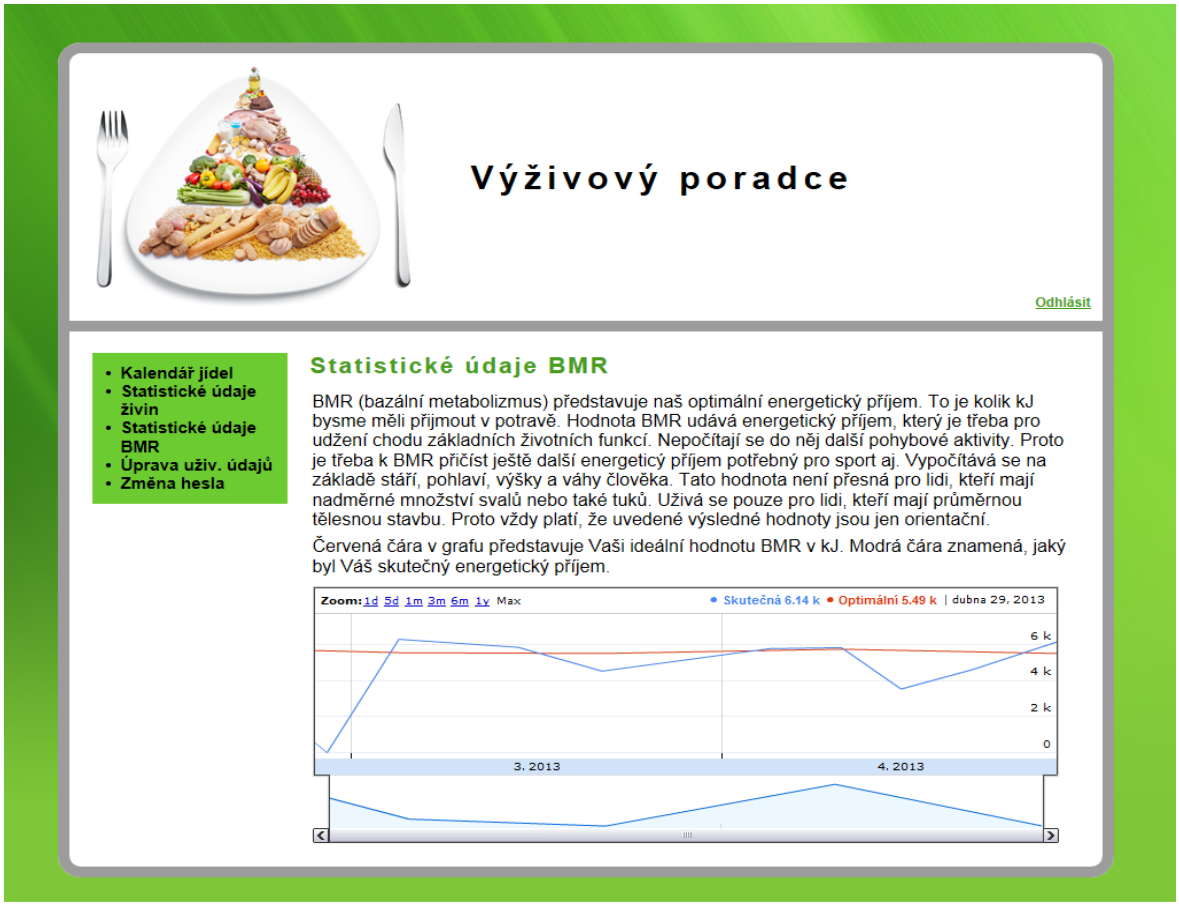


Obrázek 16 – Vzhled mobilní aplikace

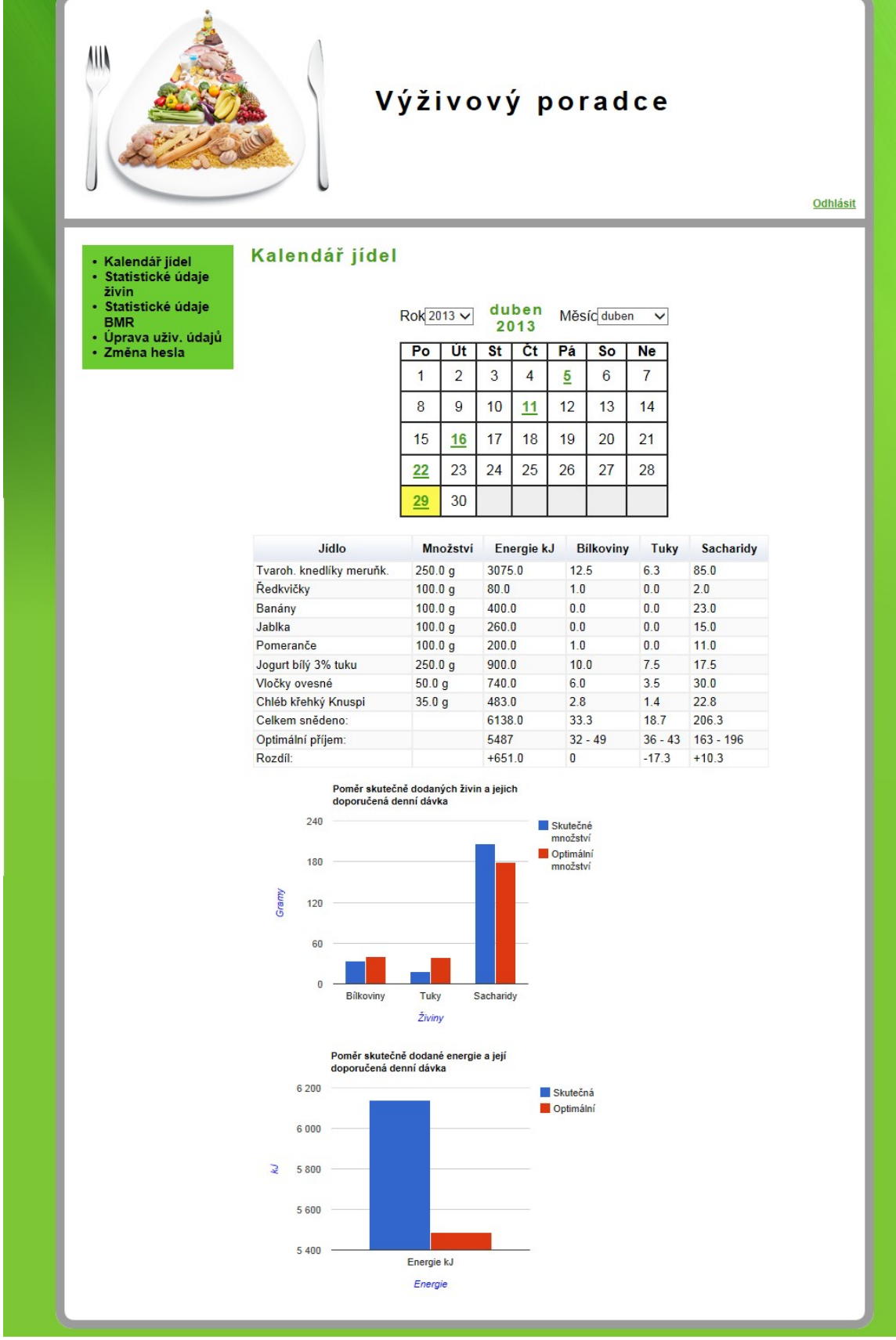
Příloha B – Vzhled webové aplikace



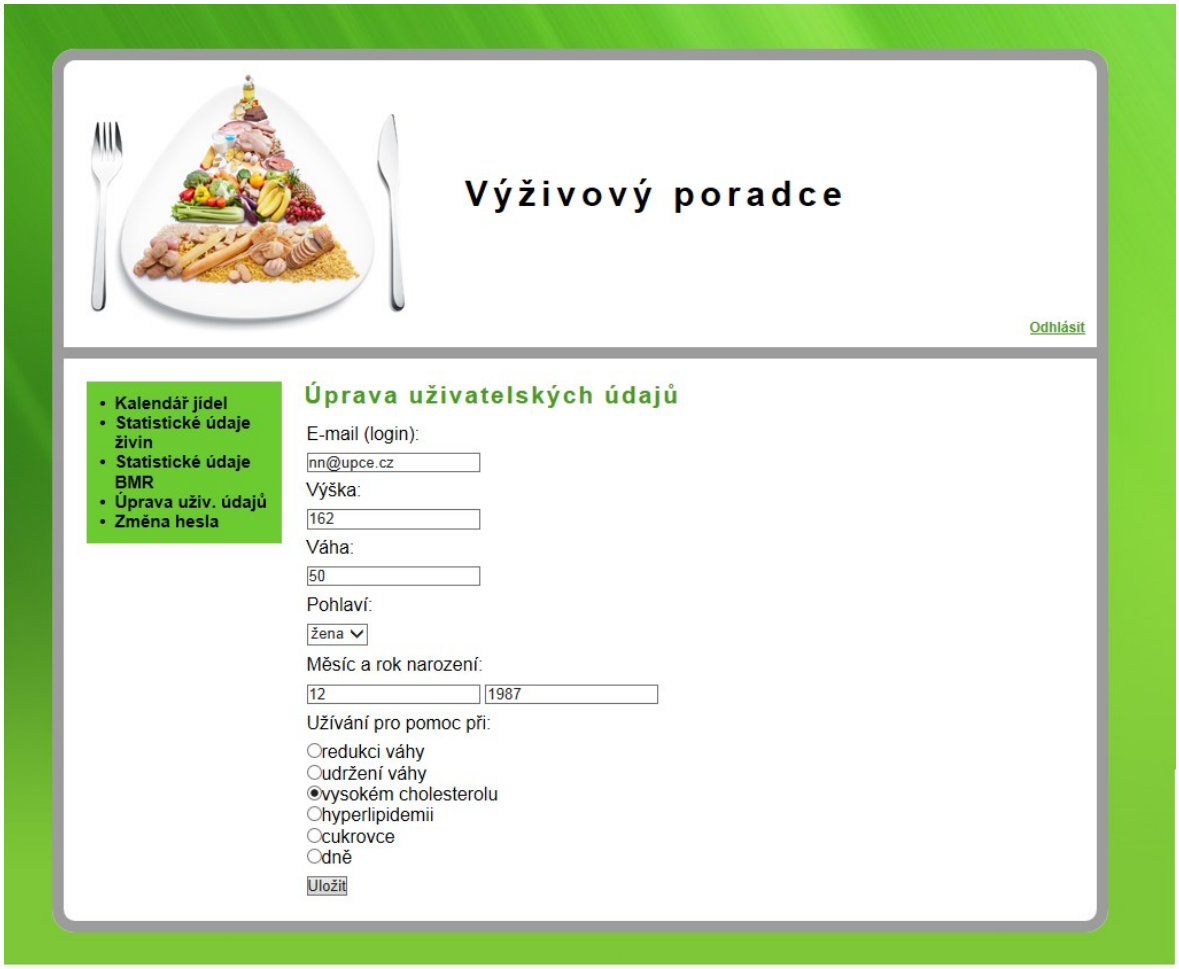
Obrázek 17 – Přihlášení ve webové aplikaci



Obrázek 18 – Statistiky webové aplikace



Obrázek 19 – Kalendář jídel webové aplikace



Obrázek 20 – Úprava uživ. údajů ve webové aplikaci